PHYS 410 Computational Physics Notes

Tobias Faehndrich

This document was last edited on April 5, 2024

Introduction:

Notes written while following the upper-level undergraduate course taught at the University of British Columbia in the Fall 2023 semester by Dr. Matt Choptuik. If any errors are found in the notes, feel free to email me at tobias.faehndrich@gmail.com. Overleaf formatting was copied from Rio W.

Contents

1 Syllabus **4**

2 Programming Basics **5**

3 Floating Point Arithmetic and Stability **6**

4 Polynomial Interpolation **8**
- 4.1 Lagrange Interpolation . . . 8
- 4.2 Barycentric Interpolation . . . 9
- 4.3 Numerical Example (EXAM HINT) . . . 9
- 4.4 Symbolic Example . . . 10

5 Solution of non-linear equations (Root finding) **11**
- 5.1 Solving nonlinear equations in one variable . . . 11
  - 5.1.1 Preliminaries . . . 11
  - 5.1.2 Bisection . . . 13

6 Newton's Method **15**
- 6.1 Newton's Method for Systems of equations . . . 16

7 Introduction to Finite Difference Approximation (FDA) **19**
- 7.1 Basic Idea via simple example . . . 19
- 7.2 Mathematical Preliminaries . . . 19
  - 7.2.1 Big-O Notation . . . 19
  - 7.2.2 Taylor Series . . . 20
  - 7.2.3 Discretization: Step 1 : Finite Difference Grids (Meshes, Lattices) . . . 20
  - 7.2.4 Discretization: Step 2 - derivation of FDAs . . . 21
- 7.3 Deriving FDAs using taylor series (exam hint) . . . 23
- 7.4 Canonical method for specifying grid spacing (nuts and bolts topic) . . . 24
- 7.5 Richardson Extrapolation . . . 25

8 The Nonlinear Pendulum **26**
- 8.1 Physical and Mathematical Formulation . . . 26
  - 8.1.1 Derivation of equation of motion . . . 26
  - 8.1.2 Non-dimensionalization . . . 27
  - 8.1.3 Linear Limit . . . 28
- 8.2 Solution via FDA . . . 28
  - 8.2.1 Discretization: Step 1 - finite difference grid . . . 28

8.2.2 Discretization Step 2 - Derivation of FDAs . . . 29
8.3 Convergence Analysis (error analysis) . . . 30
8.3.1 Convergence Test . . . 30

9 Solving Ordinary Differential Equations (ODEs) 34
9.1 Casting Systems of ODEs in first-order form . . . 34
9.2 Boundary/ Initial Conditions . . . 35
9.2.1 Initial Value Problems . . . 35
9.2.2 (Two Point) Boundary Value Problems . . . 35
9.3 Solution of ODEs (Initial Value Problems): Basic Methods . . . 35
9.3.1 Euler Methods . . . 35
9.3.2 Improving Euler Methods . . . 37
9.3.3 Runge Kutta Methods . . . 37
9.3.4 Adaptive Stepsizes . . . 38
9.3.5 Error Control with dual order method . . . 38
9.4 Checking/ Validating results from ODE integrators . . . 39
9.4.1 Monitoring conserved quantities . . . 39
9.4.2 Independent residual evaluation . . . 39
9.4.3 Simple Harmonic Motion Example . . . 41
9.5 Passing Additional Arguments to the Derivatives Function . . . 42
9.5.1 Use an anonymous function definition . . . 42
9.5.2 Use a Global Variable . . . 42
9.5.3 Adjoin parameter to ODE system . . . 43
9.6 The one dimensional Toda Lattice . . . 43
9.6.1 Analysis Methods . . . 45
9.7 Boundary Value Problems - Shooting . . . 47

10 Basic Finite Difference Techniques for Time Dependent PDEs 49
10.1 Type of IVP (By example, 1 space fim and 1 time) . . . 49
10.1.1 Wave and Wave-like ("hyperbolic equations"): The 1-d wave equation . . . 49
10.1.2 Diffusion ("Parabolic"): The 1-d diffusion equation . . . 49
10.1.3 Schrodinger: The 1-d Schrodinger equation . . . 50
10.2 Basic Concepts: Definitions (mostly review) . . . 50
10.2.1 Differential equation (ignoring BC's) . . . 50
10.2.2 Difference equation (assume characterized by single discretization scale) . . . 50
10.2.3 Residual . . . 50
10.2.4 Truncation error (don't confuse with solution error) . . . 50
10.2.5 Convergence . . . 50
10.2.6 Consistency . . . 51
10.2.7 Accuracy . . . 51
10.2.8 Solution Error . . . 51
10.3 Sample Discretizations (FDAs): 1-d Differential Equation . . . 51
10.4 Stability Analysis . . . 54
10.5 Von-Neumann (Fourier) Stability Analysis . . . 55
10.6 Some other discretizations of the diffusion equation . . . 57
10.6.1 Implicit First-Order . . . 57
10.6.2 Crank-Nicholson Scheme . . . 60
10.6.3 Exact Solution for Testing . . . 61
10.7 The 1-D Schrödinger Equation . . . 62
10.8 Wave-equation . . . 64
10.8.1 1-D Wave equation with fixed (dirichlet) B.C.'s . . . 64
10.9 Dispersion in FDAs . . . 68

11 Solution of problems in two space dimensions **70**
11.1 Diffusion Equation . . . 70
11.2 The Crank-Nicholson Scheme For 2D Diffusion Equation . . . 71
11.3 Altenating direction implicit method for the diffusion equation . . . 73
11.4 Time-Independent: Solving Elliptic PDEs . . . 74
11.5 Model Problem . . . 75
11.6 Convergence of Relaxation Methods . . . 77
11.7 Test Solution . . . 78
11.8 Successive Over Relaxation (SOR) . . . 79

12 Stochastic (random) methods **80**
12.1 Overview . . . 80
12.2 Pseudo-Random Number Generation (uniform) . . . 80
12.3 Pseudo-Random Number Generation (non-uniform) . . . 81
12.4 Diffusion Limited Aggregation (DLA) . . . 83
12.5 Physical Behaviour of Model . . . 84
12.6 Behaviour of Model for $H \neq 0$ . . . 86
12.7 Phase Transitions . . . 87

13 Fourier Transform **88**
13.1 Quick Review . . . 88
13.2 FT of Discretely Sampled Data . . . 88
13.2.1 Sampling Theorem and Aliasing . . . 88
13.2.2 Discrete Fourier Transform . . . 89
13.2.3 Fast Fourier Transform . . . 90

1 Syllabus

2 Programming Basics

Relational and Logical Operators:

- Matlab follows C approach
- 0 for false and 1 for true (for evaluating expressions)
- Relational: tilda equal (not equal), and then normal ones like ¿, ¡, etc...
- Logical: and symbol, — or symbol, tilda is not, and then double and symbol is short circuit, and double — is short circuit OR

Control Structures
Selection/Conditional:

- if-elseif-else-end statements
- no colons
- end conditional with end

Iteration (repetition, and loops)
FOR

- for ¡var¿ = ¡first¿ : ¡step¿ : ¡last¿ do something and then end
- without step, auto is step = 1
- can also give vector to iterate through or linspace
- again need to end

WHILE

- While the expression is true keep evaluating code block and then end when it is False
- can use break to get out of the innermost loop, if it is in the general code block, it will stop the running of the script

CONTINUE

- used within a loop to short circuit the execution of the loop body and proceed to the next iteration

RETURN

- causes an immediate return of a script or function to the invoking environment

Programming Units: Scripts and Functions

- in linux command you can call a script or a function script with the .m extension
- arb num of inputs and outputs
- function name(args...) for 0 output
- function output = name(args...) for 1 outputs
- function [out1, out2 .... outn] = name(in1, in2, ... inm) for n and m outputs and inputs
- Ideally name of function is the same as the name of the script

3 Floating Point Arithmetic and Stability

1. Floating Point Arithmetic

1.1 Some Definitions

- To start

$$a = \text{some exact (reference) value}$$

$$\hat{a} = \text{some approx of a (floating point of a)}$$

- Absolute Error (error)

$$e_{ABS} = a - \hat{a} \text{ or } \hat{a} - a, \text{ doesn't matter}$$

- Relative Error

$$e_{REL} = \frac{a - \hat{a}}{|a|} \text{ For } a \neq 0$$

- Note this difference is important for HW1

1.2 IEEE 64-bit floating point arithmetic

- 53 bits mantissa, II bits exponents, base-2 representation sign-bits for both mantissa / exponents.
- numbers in the range around 2×10^{-300} to $1 \times 10^{+300}$ with approx 16 decimal digits precision.
- $\frac{1}{f_{min}} < f_{MAX}$ So we don't have overflow with respect to reciprocal
- Special (exceptional) values:
  - infinity (ex 1.0/0.0)
  - nan (not a number) (ex. 0.0/0.0)
    - nan is contagious – nan (* / + -) makes anything equal nan

1.3 Machine Precision / Machine ϵ

- Smallest $\epsilon > 0$ such that

$$1 + \epsilon \neq 1 \qquad \epsilon \approx 10^{-8}$$

  In the floating point model.

1.4 Round Off error: Catastrophic loss of precision

- Multiplies/divides: relatively benign, relative error grow like $\sqrt{n}$ where n is the number of operations
- addition/subtraction: serious problem when subtracting x/y with $x \approx y$
- for example 3.6326-3.6325=0.0001
-

$$\frac{f(x+h) - f(x)}{h}$$

  as h approaches 0
- plot ln(abs error) vs —ln(h)— and we see when h is not 0 there is a catastrophic loss of precision.

1.5 Stability / Numerical Stability

- Consider some process / algorithm
$$u(x) \to y$$
where x is the continuum variables
- Floating point representation
$$\hat{u}(\hat{x}) \to \hat{y}$$
m,n component vectors
- Heuristic (non-rigorous) definition of stability:
- **Continuum**: If $|x - x'|$ is small, then so is $||u(x) - u(x')||$
- **Floating Point**: If $||\hat{x}\hat{x}'||$ is small, then so is $||\hat{u}(\hat{x}') - \hat{u}(\hat{x}||$
- Unfortunate fact: stable condition process does not imply stable floating point one.
- $\hat{u}$ not stable $\to$ Floating point finite precision will cause eros to rapidly accumulate, numerical process useless
- numerical unstable – ill-conditioned

4 Polynomial Interpolation

4.1 Lagrange Interpolation

- Notation: $(x_j, y_j) \equiv (x_j, f(x_j)) \equiv (x_j, f_j)$
- Goal: Given n Data points $(x_j, f_j) \qquad j = 1, 2, ..., n$ find (construct) unique polynomial of (maximum) degree $n-1$ passing through all data points. (Degree is the largest power of independent variable).

$$p(x) = \sum_{i=0}^{n-1} c_i x^i \qquad \text{where c is coefficients}$$

This is a polynomial of (maximum) degree $n-1$

- Once $p(x)$ is determined can use to get approximate (interpolated) values of $(f(x))$ for $x_1 \leq x \leq x_n$
- Many representations of interpolating polynomial. Lagrange approach is to take:

$$p(x) = \sum_{j=1}^{n} f_j l_j(x) \qquad \text{where the l term is of degree n-1}$$

Where $l_j(x)$ are characteristic polynomial satisfying

$$l_j(x_i) = \delta_{ji}$$

Recall the $\delta_{ji} \equiv$ Kroenicker delta where it equals 1 if $i = j$ else it equals 0

- Then

$$p(x_i) = \sum_{j=1}^{n} \frac{f_j l_i(x)}{\delta_{ji}} = \sum_{j=1}^{n} f_j \delta_{ji} = f_i$$

- Building characteristic polynomial

$$l_j(x) = \frac{(x-x_1)(x-x_2)...(x-x_{j-1})(x-x_{j+1})...(x-x_n)}{(x_j-x_1)(x_j-2)...(x_j-x_{j-1})(x_j-x_{j+1})...(x_j-x_n)}$$

- Consider $l_j(x_i) = \delta_{ji}$

$$i = j \rightarrow \text{numerator} = \text{denominator} \qquad l_j(x_j) = 1$$

$$i \neq j \rightarrow \text{exactly one term in numerator vanishes, so } l_i(x_i) = 0$$

TheoremFinal Formula: Lagrange Interpolating Polynomial

$$p(x) = \sum_{j=1}^{n} f_j l_j(x) \tag{4.1}$$

$$l_j(x) = \prod_{i=1, i\neq j}^{n} \frac{(x-x_i)}{(x_j-x_i)} \qquad n^2 \text{ number of calculations} \tag{4.2}$$

4.2 Barycentric Interpolation

Related to problem of determining centre of mass for a group of particles with given positions and weights.

- First, define $l(x)$
$$l(x) = (x - x_1)(x - x_2)...(x - x_j)...(x - x_n)$$
Then note that numerator at l_j in (2) is $l(x)/(x - x_j)$
- Next, define Barycentric weights by
$$w_j = \frac{1}{\prod_{i=1, i\neq j}^{n}(x_j - z_i)}$$
- Then Characteristic Polynomial $l_j(x)$ is $l_j(x) = l(x)\frac{w_j}{x - x_j}$
- Get the first formula for:

TheoremBarycentric Interpolation

$$p(x) = f(x) \qquad \sum_{j=1}^{n} \frac{w_j}{x - x_j} f_j \qquad \text{n number of operations} \tag{4.3}$$

- Advantage
  - Original $O(n^2)$ calculations to eliminate at any point.
  - Barycentric: $O(n^2)$ to compute weights, $O(n)$ to evaluate
- Another version: let all $f_j = 1$ then from (1) AND (3)
$$1 = \sum_{j=1}^{n} l_j(x) = l(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j}$$
- Divide (3) by last result, cancel $l(x)$ Term

TheoremBarycentric Interpolation Version 2

$$p(x) = \sum_{j=1}^{n} \frac{w_j}{x - x_j} f_j / \sum_{j=1}^{n} \frac{w_j}{x - x_j} \tag{4.4}$$

4.3 Numerical Example (EXAM HINT)

!!

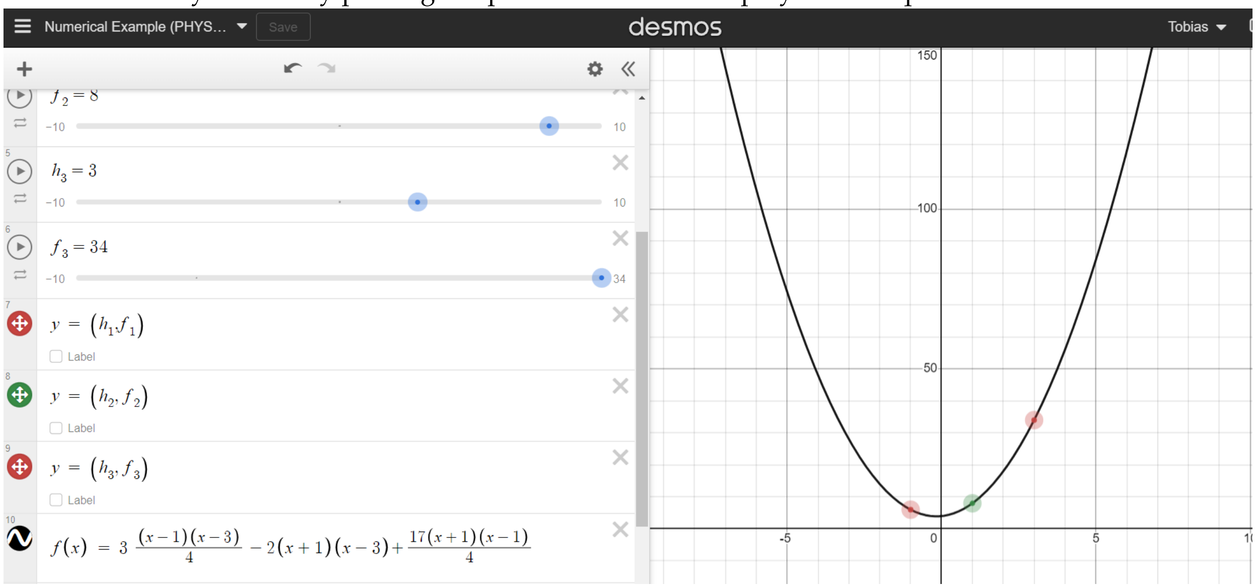
- Use (1), (2) to construct degree-2 Polynomial passing through (-1,6), (1,8), (3,34) for
$$p(x) = \sum_{j=1}^{n} f_j l_j(x)$$
$$l_j(x) = \prod_{i=1, i\neq j}^{n} \frac{(x - x_i)}{(x_j - x_i)} \qquad n^2 \text{ number of calculations}$$

$$p(x) = 6\frac{(x-1)(x-3)}{(-1-1)(-1-3)} + 8\frac{(x+1)(x-3)}{(1+1)(1-3)} + 34\frac{(x+1)(x-1)}{(3+1)(3-1)}$$

$$p(x) = 6\frac{(x-1)(x-3)}{8} + 8\frac{(x+1)(x-3)}{-4} + 34\frac{(x+1)(x-1)}{8}$$

$$p(x) = 3\frac{(x-1)(x-3)}{4} + -2(x+1)(x-3) + 17\frac{(x+1)(x-1)}{4}$$

Let me check my answer by plotting the points and the result polynomial equation:



!!!

4.4 Symbolic Example

- Consider 3 equi-spaced data points:

$$(-h, f_{-1}), (0, f_0), (+h, f_1)$$

- Using (1), (2), Construct a lagrange interpolating polynomial and evaluate it's derivative at x=0

$$\begin{aligned} p(x) &= \sum_{j=1}^{3} f_j l_j(x) \\ &= f_{-1}\frac{x(x-h)}{(-h)(-2h)} + f_0\frac{(x+h)(x-h)}{(h)(-h)} + f_1\frac{(x+h)(x)}{(2h)(h)} \\ &= f_{-1}\frac{x^2-hx}{2h^2} - f_0\frac{x^2-h^2}{h^2} + f_1\frac{x^2+hx}{2h^2} \end{aligned}$$

$$\left.\frac{d(p(x))}{dx}\right|_{x=0} = \frac{f_1 - f_{-1}}{2h} \rightarrow \text{Finite difference approximation for } f'(x)$$

5 Solution of non-linear equations (Root finding)

- Consider two cases

  1. $$f(x) = 0 \qquad \text{1-D}$$

  2. $$f(x) = 0 \qquad \text{d-D}$$

5.1 Solving nonlinear equations in one variable

- Two techniques: both presuppose info about location of root.

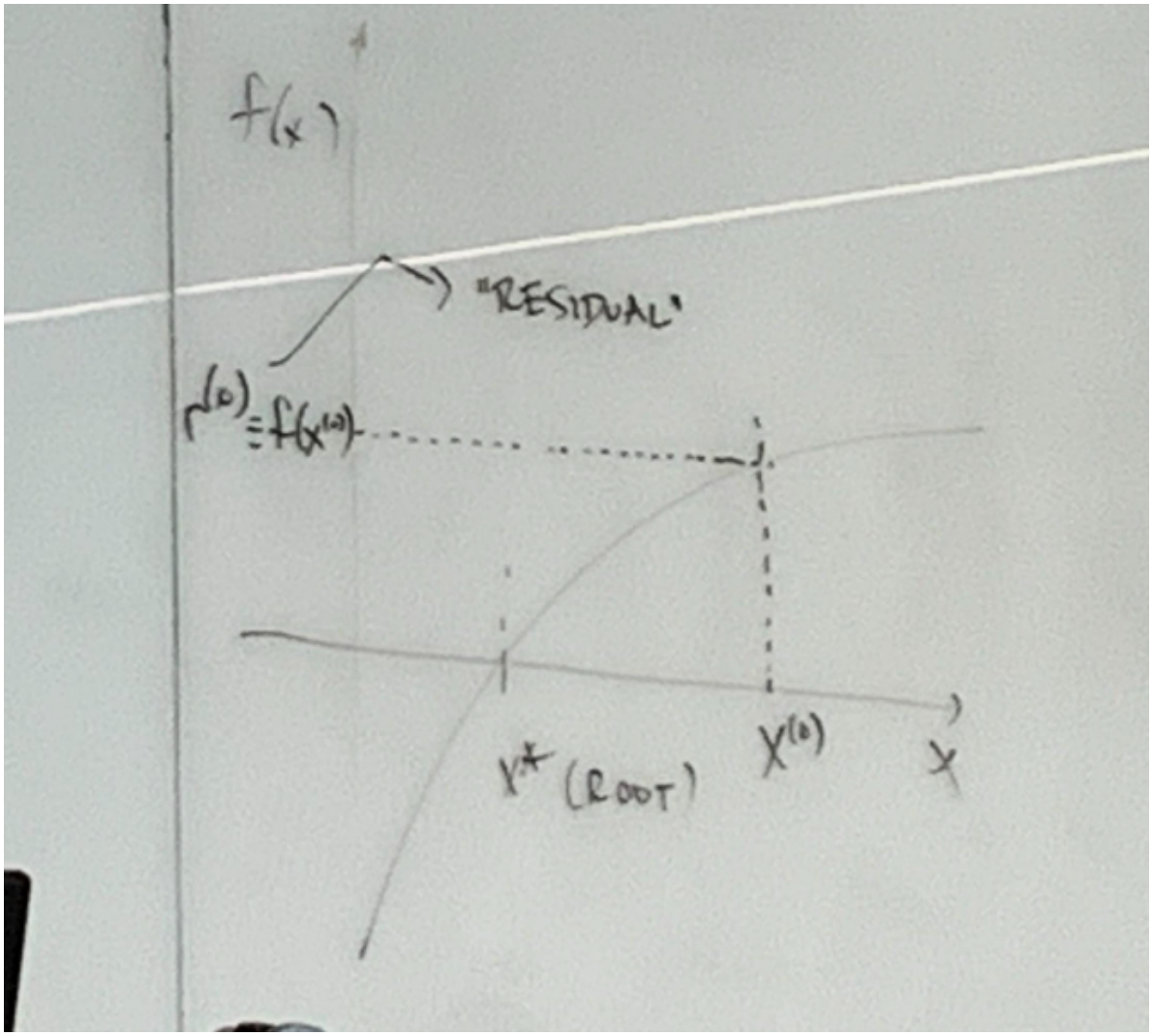
  1. Bisection (Binary search)
  2. Newton's Method (Newton-Raphson)

5.1.1 Preliminaries

- Want to find one or more roots of

$$f(x) = 0 \tag{5.1}$$

ANY nonlinear equation can be cast in this **Canonical** form



Definition: Given Canonical equation $f(x) = 0$, residual of that equation for any value x is simply $r = f(x)$

- Iterative technique

$$x^{(0)} \to x^{(1)} \to ... \to x^{(n)} \to x^{(n+1)} \to ... \to x^{(\infty)} = x^*$$

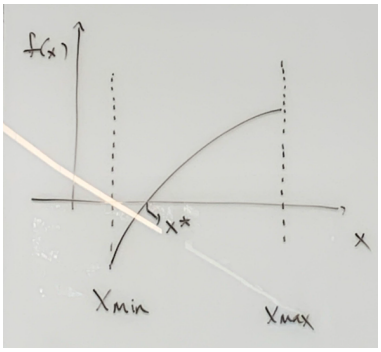
$$f(x^{(0)}) \to ... \to 0 = 0$$

$$r^{(0)} \to ... \to 0 = 0$$

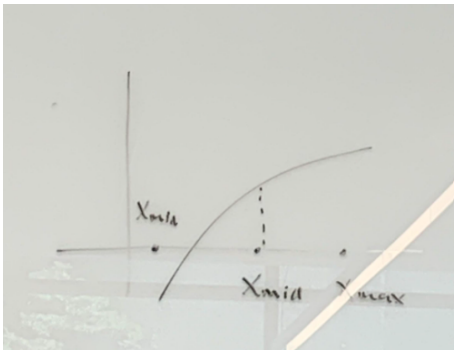
- **Convergence:** Driving residual to $0 \equiv$ Locating a root, x^*.
- Convergence: When do we stope iteration?

- Stop when
$$|\delta x^{(n)}| \equiv |x^{(n+1)} - x^{(n)}| \leq \epsilon \tag{5.2}$$
for ϵ is some prescribed convergence criterion
- Better idea: Use relativized estimate
$$\frac{|\delta x^{(n)}|}{|x^{(n+1)} \leq \epsilon|} \tag{5.3}$$
- Switch over to absolute tolerance if $|x^{(n+1)}|$ becomes "too small"
- Motivation: Small numbers often result from "unstable" processes

5.1.2 Bisection



- **Demand** that $[x_{min}, w_{max}]$ contains precisely one root
- Demand that
$$f(x_{min})f(x_{max}) < 0$$
- Start with initial bracket $[x_{min}, x_{max}] \rightarrow \delta x_0 = x_{max} - x_{min}$
Generate a series of brackets with widths
$$\delta x_1 = \delta x_0/2$$
$$\delta x_2 = \delta x_0/4$$
...
- Pseudocode (pseudo-MATLAB) - Can use for HW but cite
  - converged = false fmin = f(xmin) while not converged do xmid = (xmin+xmax)/2 fmid = f(xmid) if fmid == 0 converged = true elseif fmid*fmin ¡ 0 xmax = xmid else xmin = xmid fmin = fmid end if
  if (xmax-xmin)/abs(xmid) ¡ epsilon converge = true end if
  end while rout = xmid



- After the l-th bisection the width of the search interval is

$$\delta x_l = \delta x_0 2^{-l} = \frac{x_{max} - x_{min}}{2^l}$$

This is the upper bound on error in the root estimate

This is called divide and conquer!

6 Newton's Method

- Requires 'good' initial guess $x^{(0)}$, where the number in exponent is the iteration number
- 'Good' depends on the problem
- One general strategy: $x^{(0)}$ is a solution to a related problem, perhaps with a slightly different parameter, e.g.

Derivation of Newton's method (Taylor Series)
Newton's method requires derivative f'(x) pf f(x)

- Let x^* be a root of f(x)=0; taylor expand;

$$0 = f(x^*) = f(x^{(n)}) + (x^* - x^{(n)})f'(x^{(n)}) + O((x^* - x^{(n)})^2)$$

Where x^n is the current estimate

$$0 \approx f(x^{(n)}) + (x^* - x^{(n)})f'(x^{(n)})$$

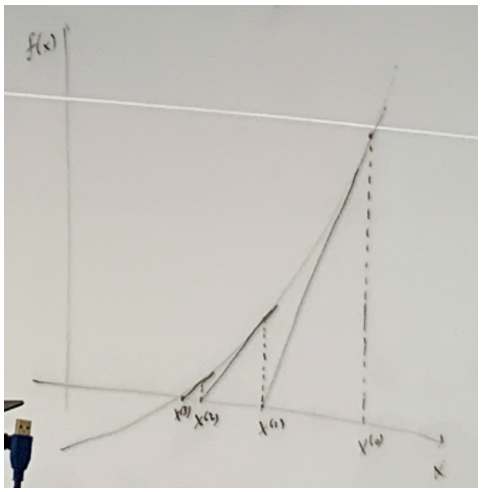
Treat as equation; let $x^* \to x^{(n+1)}$

$$0 \approx f(x^{(n)}) + (x^{(n+1)} - x^{(n)})f'(x^{(n)})$$

Or

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)}}{f'(x^{(n)})} \tag{6.1}$$

Where the denominator needs to be not equal to 0



- Can write iteration as

$$x^{(n+1)} = x^{(n)} - \delta x^{(n)} \tag{6.2}$$

$$f'(x^{(n)})\delta x^{(n)} = r^{(n)} = f(x^{(n)}) \tag{6.3}$$

- Note that we have

$$f'(x^{(n)}) = \frac{f(x^{(n)}}{x^{(n)} - x^{(n+1)}} = \frac{\text{rise}}{\text{run}}$$

Convergence: When newton's method converges, does so rapidly, expect number of sig figs to roughly double at each iteration (quadratic convergence).

Example: "Square Roots"

$$f(x) = x^2 - a = 0 \qquad \rightarrow \qquad x^* = \sqrt{a} \tag{6.4}$$

Use other equation

$$x^{(n+1)} = x^{(n)} - \frac{x^{(n)^2} - a}{2x^{(n)}}$$

$$\frac{2x^{(n)^2} - (x^{(n)^2} - a)}{2x^{(n)}} = \frac{x^{(n)^2} + a}{2x^{(n)}} = \frac{1}{2}(x^{(n)} + \frac{a}{x^{(n)}} \tag{6.5}$$

- Try it with a = 2, 12 digit arithmetic Iterate

$$x^{(0)} = 1.5$$

$$x^{(1)} = \frac{1}{2}(1.5 + \frac{2}{1.5}) = 1.41666666667 \qquad \rightarrow 3 \text{ sig figs}$$

etc...

6.1 Newton's Method for Systems of equations

- Want to solve

$$f(x) = 0$$

Where x and f are vectors

$$x = (x_1, x_2, ..., x_d)^T$$

$$f = (f_1(x), f_2(x), ..., f_d(x))^T$$

Note: Won't always show transpose

Example (d=2)

$$\sin(xy) = \frac{1}{2}$$

$$y^2 = 6x + 2$$

Ignore fact that we can reduce to d=1 case
Canonical Notation

$$\vec{x} = (x, y)$$

$$\vec{f} = (f_1(x,y), f_2(x,y))$$

$$f_1(\vec{x}) = f_1(x,y) = \sin(xy) - \frac{1}{2}$$

$$f_2(\vec{x}) = f_2(x,y) = y^2 - 6x - 2$$

- Method again iterate, start from $\vec{x}^{(0)}$
- **Note**: Determining good $x^{(0)}$ more difficult / important than 1-D case; Use continuation as necessary
- Residual Vector: $\vec{r}^{(n)}$

$$\vec{r}^{(n)} = \vec{f}(\vec{x}^{(n)}) \tag{6.6}$$

- Analog of f'(x) is Jacobian matric J of first derivatives; elements J_{ij};

$$J_{ij} = \frac{\partial f_i}{\partial x_j} \tag{6.7}$$

- Current example

$$f_1(x,y) = sin(xy) - \frac{1}{2}$$

$$f_2(x,y) = y^2 - 6x - 2$$

$$\underline{\underline{\mathbf{J}}} = \begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} \end{bmatrix} = \begin{bmatrix} y\cos xy & x\cos xy \\ -6 & 2y \end{bmatrix}$$

Newton's Method For Systems of Equations (Continued)

- Derive newton iteration: Multivariate Taylor Series Expansion

$$0 = f(x^*) = f(x^{(n)} + \underline{\underline{J}}[x^{(n)}] \cdot (x^* - x^{(n)}) + O((x^* - x^{(n)})^2)$$

  Withe Notation: $\underline{\underline{J}}[x^{(n)}] \rightarrow$ Evaluate all Jij's at $x = x^{(n)}$

- Drop higher order terms; $x^* \rightarrow x^{(n+1)}$

$$0 = f(x^{(n)} + \underline{\underline{J}}[x^{(n)}] \cdot (x^* - x^{(n)})$$

$$\text{Define } \delta x^{(n)} \equiv -(x^{(n+1)} - x^{(n)})$$

- d-dimensional newton iteration is

$$\underline{x}^{(n+1)} = \underline{x}^{(n)} - \underline{\delta x}^{(n)} \tag{6.8}$$

Where update vector $\underline{\delta x}^{(n)}$ satisfies dxd linear system

$$\underline{\underline{J}}[\underline{x}^{(n)}] \cdot \delta \underline{x}^{(n)} = \underline{f}(\underline{x}^{(n)}) \tag{6.9}$$

$\rightarrow$ Can be solved in MATLAB using left division operator

$$J_{ij}[\underline{x}^{(n)}] = \left.\frac{\partial f_i}{\partial x_j}\right|_{\underline{x}=\underline{x}^{(n)}} \tag{6.10}$$

General Structure of Newton Solver (PSEUDO CODE HW1!)

x: Solution vector

res: Residual vector

$\underline{\underline{J}}$ = Jacobian Matrix

dx: Update vector

neq: number of equations

$$||v||_2 = rms(v) = \sqrt{\frac{\sum_{i=1}^{n} v_i^2}{n}}$$

```
% x =x^(0)
% while ||dx||_2 > epsilon
%     for i = 1:new
%         res(i) = fi(x)
%         for j = 1:neq
%             J(i,j) = [dfi/dxj](x)
%         end
%     end
%     dx = J \ res % (Linear solve)
%     x = x -dx
% end
```

- Can optimize in MATLAB to not use 2 for loops.
- Here and in HW (!!!) should modify above to include code to limit the number of iterations taken (you choose that Number).

7 Introduction to Finite Difference Approximation (FDA)

7.1 Basic Idea via simple example

- Suppose We are given velocity of particle

$$v(t) \equiv \frac{dx}{dt}$$

- Can approximate velocity using

$$v(t) \equiv \frac{dx}{dt} \approx \frac{x(t+\Delta t) - x(t)}{\Delta t}$$

  Some finite increment in time; as $\Delta t \to 0$, approximation becomes more accurate.

- Solve for $x(t+\Delta t)$

$$x(t+\Delta t) \approx x(t) + \Delta t v(t)$$

- So, given x(0) can determine

$$x(\Delta t) = x(0) + \Delta t v(0)$$
$$x(2\Delta t) = x(\Delta t) + \Delta t v(\Delta t)$$
$$x(3\Delta t) = x(2\Delta t) + \Delta v(2\Delta t)$$
$$\ldots$$

- Illustrates 3 key steps in solving ODEs (PDEs) with FDA

  1. **Replace** Continuous Independent variable, t, with discrete values $0, \Delta t, 2\Delta t, 3\Delta t, \ldots$
  2. **Discretization of Equations: Replace derivatives with FDAs**
  3. **Solution:** Solve algebraic equations for approximate solution. Values (Here, once an initial condition x(0) is given)

7.2 Mathematical Preliminaries

7.2.1 Big-O Notation

- Definition: $f(h)$ is $O(h^p)$ (Note: h think stepsize, $\ll 1$) if and only if there are two positive real numbers M , h_0 such that

$$|f(h)| \leq M|h^p| \qquad \text{for all } h \; h_0$$

$$O(h^2) > O(h^3)$$

7.2.2 Taylor Series

- Taylor series and FDAs
  - Can be used to derive FDAs
  - Can be used to establish accuracy of FDAs (**)
- Usually T.S. to expand expressions like $f(x+h)$ (h; expansion parameter) about $f(x)$

$$f(x+h) = \sum_{n=0}^{\infty} h^n \frac{f^{(n)}(x)}{n!} \qquad \text{TS}$$

for nth derivative of f evaluated at x

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(x) + O(h^4)$$
$$f(x-h) = f(x) - hf'(x) + \frac{1}{2}h^2 f''(x) - \frac{1}{6}h^3 f'''(x) + O(h^4)$$

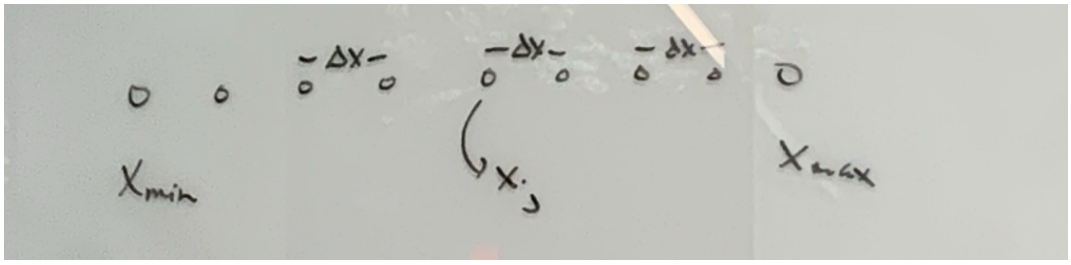
7.2.3 Discretization: Step 1 : Finite Difference Grids (Meshes, Lattices)

- Physical Domain (Continuum)

$$x_{min} \leq x \leq x_{max}$$

$$0 \leq t \leq t_{max}$$

- Uniform finite difference grid x_j



Δx is constant $\rightarrow$ uniform

- Grid function notation: $f(x_j) = f_j$
- number of grid points M_x
- Mesh spacing

$$\Delta x = \frac{x_{max} - x_{min}}{M_x - 1}$$

Grid points x_j

$$x_j = x_{min} + (j-1)\Delta x \qquad j = 1, 2, ..., M_x$$

Sept 22nd 2023

7.2.4 Discretization: Step 2 - derivation of FDAs

1. 3 for first deriv
2. 1 for 2nd deriv

- First, write down, demonstrate accuracy; later derivation
- independent variable doesn't matter (use y)

4.1 First Order, Forward FDA for 1st derivative f'(x)

$$f'(x) \approx \frac{f(x+\Delta x) - f(x)}{\Delta X}$$

Note Use a value "forward" of x.

- Grid function notation

$$f'_j \equiv f'(x_j) \approx \frac{f_{j+1} - f_j}{\Delta x}$$

$$f'(x_j) \rightarrow \frac{f_{j+1} - f_j}{\Delta x} \qquad \text{Where the arrow represents "Gets replaced with"} \tag{7.1}$$

- Accuracy?
- use taylor series $h \rightarrow \Delta x$ in (TS)

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) + \frac{1}{6}\Delta x^3 f'''(x) + O(\Delta x^4)$$

- From equation above we have

$$\frac{f(x+\Delta x) - f(x)}{\Delta x} = f'(x) + \frac{1}{2}\Delta x f''(x) + \frac{1}{6} + \Delta x^2 f'''(x) + O(\Delta x^3)$$

$$= f'(x) + \frac{1}{2}\Delta x f''(x) + O(\Delta x^2)$$

Note: Leading order truncation error is the f"(x) term

$$f'(x) + O(\Delta x)$$

- Error term $O(\Delta x)$ means we have first order accurate approx. for the deriv of f(x) at x.
- i.e. as $\Delta x \rightarrow 0$, error in approximation will tend to decrease linearly in Δx; e.g. as $\Delta x \rightarrow /2$ we also get error $\rightarrow$ error/2

4.2 First Order Backward FDA for 1st derivative f'(x)

$$f'(x) \approx \frac{f(x) - f(x-\Delta x)}{\Delta x}$$

$$f'_j \equiv f'(x_j) \approx \frac{f_j - f_{j-1}}{\Delta x}$$

$$f'(x_j) \rightarrow \frac{f_j - f_{j-1}}{\Delta x} \tag{7.2}$$

- Accuracy
- Again, use Taylor Series

$$f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) - \frac{\Delta x^3}{6} f'''(x) + O(\Delta x^4)$$

put equation above into this one

$$\frac{f(x) - f(x-\Delta x)}{\Delta x} = f'(x) - \frac{1}{2}\Delta x f''(x) + O(x^2)$$

$$= f'(x) + O(x^2)$$

4.3 Second Order Centred FDA for first derivative f'(x)

- intuitively, taking average of forward, backward should increase accuracy
- try:

$$\frac{1}{2}\left(\frac{f(x+\Delta x) - f(x)}{\Delta x} + \frac{f(x) - f(x-\Delta x)}{\Delta x}\right)$$

$$= \frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x}$$

$$f'(x_j) \to \frac{f_{j+1} - f_{j-1}}{2\Delta x} \tag{7.3}$$

- Accuracy

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) + \frac{1}{6}\Delta x^3 f'''(x) + O(\Delta x^4)$$

$$f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) - \frac{1}{6}\Delta x^3 f'''(x) + O(\Delta x^4)$$

Subtract them and lots cancels out

$$\frac{f(x+\Delta x) - f(x-\Delta x)}{2\Delta x} = f'(x) + \frac{1}{6}\Delta x^3 f'''(x) + O(\Delta x^4)$$

$$= f'(x) + O(\Delta x^2)$$

Second order accuracy

- e.g. as $\Delta x \to \Delta x/2$ we find error $\to$ error$/4$
- approximation is called "centred"; structure of formula is symmetric about x_j

4.4 Second Order Centred Diff. Approx for f"(x)

- claim: Following is $O(\Delta x^2)$ approx of f''(x)

$$f''(x) \approx \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2}$$

$$f''(x_j) \rightarrow \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2} \tag{7.4}$$

- Justification: Use Taylor Series

$$f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) - \frac{\Delta x^3}{6} f'''(x) + \frac{\Delta x^4}{24} f''''(x) + O(\Delta x^5)$$

$$-2f(x) = -2f(x)$$

$$f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{\Delta x^2}{2} f''(x) + \frac{\Delta x^3}{6} f'''(x) + \frac{\Delta x^4}{24} f''''(x) + O(\Delta x^5)$$

Subtract terms (reverse engineering style to try to get back to claimed equation)

$$\begin{aligned} \frac{f(x+\Delta x) - 2f(x) + f(x-\Delta x)}{\Delta x^2} &= f''(x) + \frac{\Delta x^2}{12} f''''(x) + O(\Delta x^4) \\ &= f''(x) + O(\Delta x^2) \end{aligned}$$

Second order error term (as claimed)

7.3 Deriving FDAs using taylor series (exam hint)

- Need to choose points that will be used in FDA

. . . . . .

x_{j-1}, x_j, x_{j+1}

- Example: Determine approximation to f''(x) that uses x_j, x_{j+1}, x_{j-1}
- Assume that an appropriate linear combination of truncated taylor series for f_{j-1}, f_j, f_{j+1} will give the formula

$\rightarrow$ consider:

$$\alpha f_{j-1} + \beta f_j + \gamma f_{j+1} = f''(x) + ... \tag{7.5}$$

- Taylor expanding

$$f_{j-1} = f(x-\Delta x) = f(x) - \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) - \frac{\Delta x^3}{6} f'''(x) + O(\Delta x^4)$$

$$f_j = f(x)$$

$$f_{j+1} = f(x+\Delta x) = f(x) + \Delta x f'(x) + \frac{1}{2}\Delta x^2 f''(x) + \frac{\Delta x^3}{6} f'''(x) + O(\Delta x^4)$$

- Now require equation give f″(x) at leading order

$$\alpha + \beta + \gamma = 0 \qquad \text{(No f(x) term)}$$
$$-\alpha + \gamma = 0 \qquad \text{(No f′(x) term)}$$
$$\frac{\Delta x^2}{2}(\alpha + \gamma) = 1 \qquad \text{(Gives us f″(x))}$$

- Solving

$$\alpha = \frac{1}{\Delta x^2}$$

$$\beta = \frac{-2}{\Delta x^2}$$

$$\gamma = \frac{1}{\Delta x^2}$$

- So our FDA is

$$f''(x_j) \rightarrow \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2}$$

problem hint for exam

7.4 Canonical method for specifying grid spacing (nuts and bolts topic)

. . . . . .

Mesh (partial) grid with each dot separated by Δx

Nothing special about spacing, so we can change it if we want...?

- Whenever solving DE's with FDA, change Δx to see what happens
- Changes by a factor of 2 are most convenient

$$\Delta x, \Delta x/2, \Delta x/4, \Delta x/8...$$

$$l, l+1, l+2, l+3$$

- Discretization parameter: Level l

$$n_x = 2^l + 1$$

$$\Delta x = \frac{x_{max} - x_{min}}{2^l}$$

  Just easier to think/specify in terms of l.

- Very small values of l not likely to be of much use; more likely to use $l = 6, 7, 8, ...$ corresponding to $n_x = 65, 129, 257, ...$
- As l and n_x get larger, Δx gets smaller and FDA should become more accurate.

7.5 Richardson Extrapolation

- Basic Idea: FD approximation using different scales of discretization $\Delta x_1, \Delta x_2, \Delta x_3, ...$ but the same F.D. template $\rightarrow$ combine to get higher order approximation.
- Example: Forward approx. of first derivative
- Recall

$$L^{\Delta x} f = \frac{f(x+\Delta x) + f(x)}{\Delta x} = f'(x) + \frac{1}{2}\Delta x f''(x) + \frac{1}{6}\Delta x^2 f'''(x) + O(\Delta x^3)$$

$$L^{2\Delta x} f = \frac{f(x+2\Delta x) + f(x)}{2\Delta x} = f'(x) + \frac{1}{2}2\Delta x f''(x) + \frac{1}{6}2\Delta x^2 f'''(x) + O(\Delta x^3)$$

- Take linear combination

$$\alpha L^{\Delta x} f + \beta l^{2\Delta x} f = f'(x) + O(\Delta x^2)$$

  So giving us

$$\alpha + \beta = 1$$

$$\alpha + 2\beta = 0$$

  ("again this sort of thing is Good for exam purposes again")

- Solve 2 equations

$$\alpha = 2$$

$$\beta = -1$$

  giving us

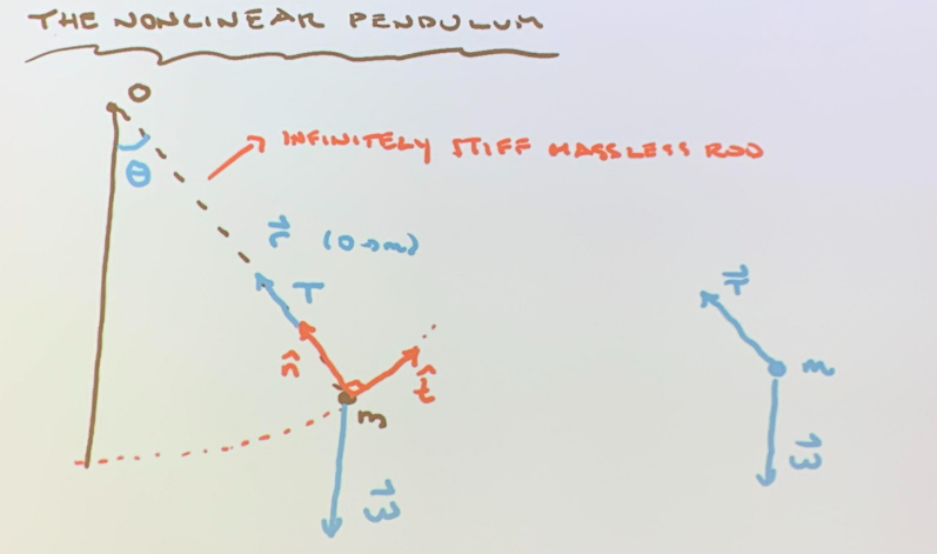
$$\alpha L^{\Delta x} f + \beta l^{2\Delta x} f = s\left(\frac{f_{j+1} - f_j}{\Delta x}\right) - \left(\frac{f_{j+2} - f_j}{2\Delta x}\right)$$

$$\rightarrow \qquad \frac{1f_{j+2} + 4f_{j+1} - 3f_j}{2\Delta x}$$

$$= f'(x) - \frac{1}{3}\Delta x^2 f'''(x) + O(\Delta x^3) = f'(x) + O(\Delta x^2)$$

- Can linear combination of 3,4,... approximations to constant even more accurate formulae

8 The Nonlinear Pendulum



8.1 Physical and Mathematical Formulation

- Mass of Bob: m
- Infinetely rigid, massless pendulum arm; Length L $\equiv |\vec{r}|$
- No friction at pivot point. 0
- total mechanical energy (kinetic and potential) conserved

8.1.1 Derivation of equation of motion

- Displacement vector $\vec{r}(t)$
- $\vec{r}(t)$ makes angle $\theta(t)$ (dynamical variable) with vertical
- Normal-tangential coordinate system, unit vectors; $\hat{n}$ and $\hat{t}$
- Velocity $\vec{v}(t)$ of Bob

$$\vec{v}(t) = \frac{d\vec{r}}{dt} = v\hat{t}$$
$$v \equiv |\vec{v}|$$

Velocity is purely tangential

- Acceleration $\vec{a}(t)$

$$\vec{a}(t) = \frac{d^2\vec{r}}{dt^2} = \frac{d\vec{v}}{dt} = \frac{d}{dt}(v\hat{t}) = \frac{dv}{dt}\hat{t} + v\frac{d\hat{t}}{dt} = \frac{dv}{dt}\hat{t} + \frac{v^2}{L}\hat{n}$$

Using $\frac{d\hat{t}}{dt} = \frac{v}{L}\hat{n}$ and fact that L is constant

- no motion in normal direction; consider only tangential motion

- tension force $\vec{T} = T\hat{n}$ and normal component of weight wcost irrelevant
- newton's 2nd law:

$$ma_t = F_t = -\omega \sin\theta = -mg \sin\theta$$

where g is the acceleration due to gravity

$$a_t = -g \sin\theta$$

- rewrite as a function for $\theta(t)$
- angular velocity

$$\omega(t) = \frac{d\theta}{dt}$$

- Angular acceleration

$$\alpha(t) = \frac{d\omega}{dt} = \frac{d^2\theta}{dt^2}$$

$$L\alpha = a_t = L\frac{d^2\theta}{dt^2}$$

- sub in equation above and get:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\sin\theta \qquad 0 \leq t \leq t_{max}$$

- Need initial conditions

$$\theta(0) = \theta_0$$
$$\omega(0) = \omega_0$$

these are specified values

8.1.2 Non-dimensionalization

- Can change system of units so g=1, L=1, simplifies equation of motion; using **Natural Units** for problem.
- Adopting our new set of units, we want to solve is:

$$\frac{d^2\theta}{dt^2} = -\sin\theta \qquad 0 \leq t \leq t_{max} \tag{8.1}$$

with initial conditions

$$\theta(0) = \theta_0 \tag{8.2}$$

$$\omega(0) = \omega_0 \tag{8.3}$$

- Equation above is nonlinear; can solve in closed form but the solution is complicated; FDA solution is no more complicated for a nonlinear case than for a linear case.

8.1.3 Linear Limit

- Assume $\theta(t) \ll 1$

$$\sin\theta \approx \theta$$

(3) $\rightarrow$

$$\frac{d^2\theta}{dt^2} = -\theta \qquad 0 \leq t \leq t_{max} \tag{8.4}$$

- Has clenemal solution

$$\theta(t) = A\sin(t+\delta)$$

where natural units, $\Omega \equiv 1$
and A, δ determined by initial conditions

$$\Omega \equiv \sqrt{\frac{g}{L}} = 1$$

Independent of oscillation amplitude, A (equivalently independent of initial conditions)

- Not the case for the nonlinear pendulum

8.2 Solution via FDA

(HINT: serves as prototype for work in project 1)
Recall method:

- Illustrates 3 key steps in solving ODEs (PDEs) with FDA
  1. **Replace** Continuous Independent variable, t, with discrete values $0, \Delta t, 2\Delta t, 3\Delta t, ...$
  2. **Discretization of Equations: Replace derivatives with FDAs**
  3. **Solution:** Solve algebraic equations for approximate solution. Values (Here, once an initial condition x(0) is given)

Now Implement:

8.2.1 Discretization: Step 1 - finite difference grid

- Continuous domain:

$$0 \leq t \leq t_{max}$$

- Specify mesh via level parameter, l

$$n_t = 2^l + 1$$

$$\Delta t = \frac{t_{max}}{n_t + 1} = 2^{-l} t_{max}$$

$$t^n = (n-1)\Delta t, \qquad n = 1, 2, ..., n_t$$

8.2.2 Discretization Step 2 - Derivation of FDAs

- Continuum equations $\rightarrow$ Discrete Equations

$$\theta^n \equiv \theta(t^n) \equiv \theta((n-1)\Delta t)$$

- One derivative to replace, use $O(\Delta t^2)$ centred approximation

$$\left.\frac{d^2\theta}{dt^2}\right|_{t=t^n} \approx \frac{\theta^{n+1} - 2\theta^n + \theta^{n-1}}{\Delta t^2} \tag{8.5}$$

  - t^{n+1}
  - t^n $\rightarrow$ Centre-point for formula
  - t^{n-1}

- Should evaluate $\sin\theta$ at $t = t^n$ $\rightarrow \sin\theta^n$
- substituting in the equation above (27)

$$\frac{\theta^{n+1} - 2\theta^n + \theta^{n-1}}{\Delta t^2} = -\sin\theta^n \qquad n+1 = 3, 4, ..., n_t \tag{8.6}$$

Nonlinear pendulum (continued) September 29th 2023

- Sove for θ^{n+1}

$$\theta^{n+1} = 2\theta^n - \theta^{n-1} - \Delta t^2 \sin\theta^n \qquad n+1 = 3, 4, ..., n_t \tag{8.7}$$

- First discrete time we can use this, $t^{n+1} = t^3$
- Need value $\theta' = \theta(0)$; $\theta^2 = \theta(\Delta t)$ to initialize scheme
- θ' given by initial condition

$$\theta' = \theta(0) = \theta_0$$

- θ^2 $\rightarrow$ bit more involved; state without proof that we need θ^2 to $O(\Delta t^3)$ accuracy so that overall solution is $O(\Delta t^2)$
- Heuristic justification: $t = t_{max}$ final time number of time steps is $\approx \frac{1}{\Delta t} = O(\Delta t^{-1})$. So per time step error needs to be $O(\Delta t^3)$ so that overall solution is $O(\Delta t^2)$
- Proceed via taylor series, use initial conditions

$$\theta(\Delta t) = \theta(0) + \Delta t \frac{d\theta}{dt}(0) + \frac{1}{2}\Delta t^2 \frac{d^2\theta}{dt^2}(0) + O(\Delta t^3)$$

$$\approx \theta_0 + \Delta t\omega_0 + \frac{1}{2}\Delta t^2 \frac{d^2\theta}{dt^2}(0)$$

Use equation of motion

$$= \theta_0 + \Delta t\omega_0 - \frac{1}{2}\Delta t^2 \sin\theta_0$$

- Assembling results

$$\theta^{n+1} = 2\theta^n - \theta^{n-1} - \Delta t^2 \sin\theta^n \qquad n+1 = 3, 4, ..., n_t$$

$$\theta^1 = \theta_0$$

$$\theta^2 = \theta_0 + \Delta t\omega_0 - \frac{1}{2}\Delta t^2 \sin\theta_0$$

- Now have n_t equations in n_t unknowns and can solve.

8.3 Convergence Analysis (error analysis)

- Want to examine behaviour of solution as $\Delta t \to 0$
- Assumption (Richardson, 1909) Let $U_*(t)$ be the exact (continuum) solution of some differential equation (example the equations above in example); then error is

  $$e(t^n) \equiv U_*(t^n) - U(t^n)$$

  Where $U(t^n)$ is the computed value

  And takes the form

  $$\lim_{\Delta t \to 0} e(t^n) = \Delta t^2 e_2(t^n) + O(\Delta t^4)$$

  where e_2 term is some function not some "random" error that would be seen in analyzing experimental data. note that order is 4 because the FDA is centred

- How do we know this?
- Deep question. Our approach: is to use it as an assumption for convergence analysis.

8.3.1 Convergence Test

Some mesh:

$$o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o$$

$$l, \Delta t_l, u_l^n$$

$$o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o \quad o$$

$$l+1, \Delta t_{l+1} = \Delta t/2, u_{l+1}^n$$

$$o\ o\ o\ o\ o\ o\ o\ o\ o\ o\ o$$

$$l+2, \Delta t_{l+2} = \Delta t/4, u_{l+2}^n$$

Always use at least 3 grids/ meshes, values of l

- Then from equation above we have

$$u_l^n \approx u_*^n - (\Delta t_l)^2 e_2^n$$

$$u_{l+1}^n \approx u_*^n - (\Delta t_{l+1})^2 e_2^n$$

$$u_{l+2}^n \approx u_*^n - (\Delta t_{l+2})^2 e_2^n$$

  Think of n labelling common set of times (times on caorse gride l)

- Now subtract solutions on adjacent levels

$$u_l^n - u_{l+1}^n \approx -((\Delta t_l)^2 - (\Delta t_{l+1})^2) e_2^n = \frac{-3}{4} \Delta t_l^2 e_2^n$$

$$u_{l+1}^n - u_{l+2}^n \approx -\frac{3}{4}((\Delta t_{l+1})^2 e_2^n = \frac{-3}{16} \Delta t_l^2 e_2^n$$

  where the last term is error reduced by factor of 4

- Observation

  1. Get estimate of solution error by simply subtracting solutions on adjacent levels
     From equation earlier we have

$$e^n = \Delta t^2 e_2^n + ...$$

$$u_l^n - u_{l+1}^n \approx -\frac{3}{4}(\Delta t_l)^2 l_2^n$$

     so

$$-\frac{4}{3}(u_l^n - u_{l+1}^n) \approx e^n$$

  2. Consider the ratio

$$\frac{u_l^n - u_{l+1}^n}{u_{l+1}^n - u_{l+2}^n}$$

     in limit that $\Delta t_l \to 0$ should get

$$\frac{-(3/4)(\Delta t_l)^2 e_2^n}{-(3/16)(\Delta t_l)^2 e_2^n} = 4$$

  3. More useful in practice: scale (multiply) differences

$$u_l^n - u_{l+1}^n \text{ by } 4^0 = 1$$
$$u_{l+1}^n - u_{l+2}^n \text{ by } 4^1 = 4$$
$$u_{l+2}^n - u_{l+3}^n \text{ by } 4^2 = 16$$

     And plot them (as a function of t^n) on single graph

     Curves should be nearly coincident and agreement should get better for higher levels

4. Can use as many levels in the test as is feasible but use a minimum of 3
5. Important: don't have to know what the error is to do a convergence test
6. Important: If we don't observe convergence, good indication that we need to do some debugging.

11:10am October 4th 2023
Energy Conservation
Mechanical energy is conserved = use as check on implementation

- For nonlinear pendulum

$$\text{kinetic energy} \equiv T(t) = \frac{1}{2}mv(t)^2 = \frac{1}{2}m(L\omega(t))^2$$

$$\text{potential energy} \equiv V(t) = mgh(t)$$

where h(t) is vertical displacement of bob from equilibrium height

$$\text{total energy} = E(t) = T(t) + V(t)$$

$$\text{Total Energy } = E(t) = T(t) + V(t)$$

$$E(t) = \frac{1}{2}m(L\omega(t))^2 + mgL(1 - \cos\theta(t))$$

- Our units: g=1, L=1, m=1 (MLT)

$$E(t) = T(t) + v(t) = \frac{1}{2}\omega(t)^2 + (1 - \cos\theta(t)$$

- Track deviation in energy

$$dE(t) \equiv E(t) - E(0)$$

- One thing to do: plot it and see whether it "Looks Good"
- Better: Check convergence of $dE(t;\Delta t)$

$$\Rightarrow \text{Expect} \qquad dE(t;\Delta t) = \Delta t^2 \epsilon(t) + ... \qquad \text{Where epsilon is some function}$$

$$\Rightarrow \text{As } \Delta t \to 0 \qquad dE(t;\Delta t) \to 0 \text{ As } \Delta t^2$$

$$\Rightarrow \text{Plot} \quad dE(t;\Delta t), 4dE(t;\Delta t/2), 16dE(t;\Delta t/4), ...$$

Should be neatly coincident

- Calculations should display convergence to conservation
- For **linear** pendulum need another expression for total energy

- Small angle approximation

$$\cos\theta = 1 - \frac{1}{2}\theta^2 + O(\theta^4) \approx 1 - \frac{1}{2}\theta^2$$

$$\Rightarrow E_{Linear}(t) = T(t) + V(t) = \frac{1}{2}\omega(t)^2 + \frac{1}{2}\theta(t)^2$$

End of FDA discussion - will use convergence testing and FDA lots in projects and HW!

9 Solving Ordinary Differential Equations (ODEs)

9.1 Casting Systems of ODEs in first-order form

- Can always reduce a system of ODEs to a set of first order DEs by introducing appropriate new (auxiliary) variables Example 1:

$$y''(x) + q(x)y'(x) = r(x) \qquad ' \equiv \frac{d}{dx} \tag{9.1}$$

This is second order since the highest derivative is ″

- Introduce new variable $z(x) \equiv g'(x)$ then (30) becomes

$$y' = z$$

$$z' = r - qz$$

No derivatives on the RHS

Example 2:

$$y''''(x) = f(x) \tag{9.2}$$

- Introduce new variables

$$y_1(x) \equiv y'(x)$$
$$y_2(x) \equiv y''(x)$$
$$y_3(x) \equiv y'''(x)$$

Then (31) Becomes;

$$y' = y_1$$
$$y_1' = y_2$$
$$y_2' = y_3$$
$$y_3' = f$$

- Thus, generic problem in ODEs is reduced to study of N coupled, **first order** DEs for functions

$$y_i = 1, 2, 3, ...N$$

$$y_i'(x) \equiv \frac{dy_i}{dx}(x) = f_i(x, y_1, y_2, ..., y_N) \qquad i = 1, 2, .., N \tag{9.3}$$

Where the f_i are known functions of x, y_i

Equivalent forms

$$\underline{y}'(x) = \underline{f}(x, y)$$

$$\dot{\underline{y}}(t) = \underline{f}(t, \underline{y})$$

dot is the time derivative

Now to actually solve some ODEs we need something else...

9.2 Boundary/ Initial Conditions

- ODE problem not completely specified by ODEs themselves
- Generally, boundary conditions (BCs) are algebraic conditions on the yi in (32) that are to be setastifed at discrete specified points.
- Nth order system $\Rightarrow$ need N conditions
- BCs divide ODE problems into 2 broad classes:
  1. Initial value problem
  2. Boundary value problem

9.2.1 Initial Value Problems

- All the y_i are given at some initial value t_{min} (0), wish to find the y_i at some final value t_{max} (or set of values)

$$t_n \qquad t_{min} \leq t_n \leq t_{max} \quad n = 1, 2, ...$$

- Most relevant for recasting of ODEs in first order form

9.2.2 (Two Point) Boundary Value Problems

- BC's Specified at more than one value of x;
  typically: some at x_{min}, some at x_{max}

9.3 Solution of ODEs (Initial Value Problems): Basic Methods

9.3.1 Euler Methods

- Consider the basic ODE

$$y'(x) = f(x, y)$$

To be solved on some interval $[x_0, x]$ where $y(x_0)$ is given.

- One approach: taylor series

$$y(x) = y(x_0) + (x - x_0)y'(x_0) + \frac{(x - x_0)^2}{2}y''(x_0) + \dots$$

Where $y'(x_0) = f(x_0, y_0)$

- Higher derivatives get messy

$$y''(x) = \frac{\partial}{\partial x} f(x,y) + \frac{dy}{dx}\frac{\partial}{\partial y} f(x,y)$$
$$= \frac{\partial f}{\partial x} + f(x,y)\frac{\partial}{\partial y} f(x,y)$$

- Algebraically complicated to go to high order; not **often** used in practice; good for derivations
- Truncate expansion at $O(x - x_0)$

$$y(x) \approx y(x_0) + (x - x_0)y'(x_0)$$

$$= y(x_0) + (x - x_0)f(x_0, y_0)$$

  where the dx term is equiv to h, i.e. the step size

- $\Rightarrow$ Euler Method (forward Euler)

$$y(x_0 + h) = y(x_0) + hf(x_0, y_0) = y_0 + hf_0 \tag{9.4}$$

- Basic algorithm for any ODE integrator: take repeated steps, possibly adjusting step size until integration limit is reached.

$$y_{n+1} = y_n + hf_n$$

- Accuracy? O(h)
- Not recommended for production work
- Improve accuracy by using value of y' at mid point of the interval
- Want an estimate of $f(x_0 + h/2, y(x_0 + h/2))$
- Compute

$$y(x_{mid}) = y_0 + \frac{h}{2}y'_0 = y_0 + \frac{h}{2}f_0$$

- Modified Euler's method

$$y(x_0 + h) = y_0 + hf(x_{mid}, y_{mid})$$
$$x_{mid} = x_0 + \frac{h}{2}$$
$$y_{mid} = y_0 + \frac{h}{2}f_0$$

  this is $O(h^2)$ accurate

- Improved Euler Method

$$y(x_0 + h) = y(x_0) + hf_0 + \frac{f(x_0 + h, y_0 + hf_0)}{2}$$

  Still $O(h^2)$ accurate

9.3.2 Improving Euler Methods

- Taylor Series: use more terms in expansion
- Linear Multi-step methods: use data from previous time steps to cancel terms in truncation error
- Runge-Kutta methods: use intermediate points within time step
- So far we have been looking at:

$$y'(x) = f(x, y(x))$$

- All methods so far and to come generalize immediately to

$$y_i'(x) = \frac{dy_i}{dx}(x) = f_i(x, y, \ldots, y_N)$$

for $i = 1, 2, \ldots, N$

$$\mathbf{y}'(\mathbf{x} = \mathbf{f}(x, \mathbf{y}(\mathbf{x}))$$

9.3.3 Runge Kutta Methods

- Modified Euler Methods is an example of a runge-kutta (RK) method of order 2
- For any given order, infinite number of distinct RK methods
- Very popular example: Fourth order (RK4):

$$f_0 = f(x_0, y_0)$$
$$f_1 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} f_0)$$
$$f_2 = f(x_0 + \frac{h}{2}, y_0 + \frac{h}{2} f_1)$$
$$f_3 = f(x_0 + h, y_0 + \frac{h}{2} f_2)$$

$$y(x_0 + h) = y(x_0) + \frac{h}{6}(f_0 + 2f_1 + 2f_2 + = f_3)$$

Here we get $O(h^4)$ accuracy! Good compromise between accuracy and the number of function evaluations.

- Non-trivial to derive RK4, but for the case where $f = f(x)$, can construct it by approximating

$$y(x_0 + h) = y(x_0) + \int_{x_0}^{x_0+h} f(x)dx$$

Integral using Simpson's rule

9.3.4 Adaptive Stepsizes

- In general, want solution of ODEs to some prescribed accuracy
- What affects the solution accuracy?
  - Step size, h
  - Accuracy order of method (p)
- How do we **estimate** the solution accuracy?
  - Typically estimate **local** accuracy
  - Basic tool: compute solution at $x = x_0 + h$ using two different step sizes or methods of different accuracy and then compare them. (HW2 Hint)
- How do we control solution accuracy?
  - If difference is less than error criterion, accept solution and advance to next time step; otherwise decrease stepsize and repeat integration
  - If difference is much less than error criterion, increase step size
  - In practice implement limiters so that stepsizes don't change too much from one time step to another.

9.3.5 Error Control with dual order method

- Two methods (RK, say) with orders of accuracy p, p+1
- To leading order, after one step with each method

$$y(x_0 + h) = y_{\text{exact}} + kh^{p+1}$$

  p+1 comes from the fact that this is a local error; $O(h^{-1})$ steps to get to any finite time

$$\tilde{y}(x_0 + h) = y_{\text{exact}} + \tilde{k}h^{p+2}$$

- Assume h is small; difference between two is

$$y(x_0 + h) - \tilde{y}(x_0 + h) \approx kh^{p+1} - \tilde{k}h^{p+2} \approx kh^{p+1}$$

  $\Rightarrow$ direct estimate of the solution error
- Typical integrator will use 2 types of error control parameterized by ϵ_{ABS}, ϵ_{REL}
  - ϵ_{ABS}: Absolute error control; integrator tries to keep error estimate at/below ϵ_{ABS}
  - ϵ_{REL}: Relative error control; integrator attempts to maintain

$$\frac{e_{\text{EST}}}{|y(x_0 + h)|} \leq \epsilon_{\text{REL}}$$

- Absolute control: Good when values are O(1) or relatively constant in magnitude or near 0 (numerically ill-conditioned process)
- Relative control: good when solution magnitudes vary appreciably

- Can combine two; for example we can say

$$e_{\text{EST}} \leq \epsilon_{\text{ABS}} + \epsilon_{\text{REL}}|y(x_0 + h)|$$

or

$$e_{\text{EST}} \leq \text{MAX}(\epsilon_{\text{ABS}}, \epsilon_{\text{REL}}|y(x_0 + h)|$$

This is what MATLAB integrators tend to use

9.4 Checking/ Validating results from ODE integrators

9.4.1 Monitoring conserved quantities

- Example: for dynamical systems with a Lagrangian/ Hamiltonian, total energy E(t) is conserved
- Monitor variation $\delta\hat{E}(t;\epsilon)$ on a solution domain $0 \leq t \leq t_{max}$

$$\delta\hat{E}(t;\epsilon) = \hat{E}(t;\epsilon) - \hat{E}(0;\epsilon)$$

where $\epsilon \equiv$ error tolerance for integrator

- Should find that $\delta\hat{E}(t;\epsilon)$ is $O(\epsilon)$

$$\delta\hat{E}(t;\epsilon) = \epsilon g(t) + \ldots$$

where g(t) is some function

- Thus, e.g., if we take $\epsilon \to \epsilon/10$ should see $\delta\hat{E} \to \delta\hat{E}/10$ (approx so long as $\epsilon \gg \epsilon_{\text{machine}}$)

9.4.2 Independent residual evaluation

- **Idea**: Attempt to directly verify that approximate solution (previously y) satisfies the ODEs through the use of an independent discretization of the ODEs (distinct from the one used by the ODE integrator)
- Residual $\Rightarrow$ something that should tend to 0 in appropriate limit
- Let

$$L[u(t)] \equiv Lu(t) = 0$$

where L is a differential operator

  - u(t) can be vector of dependent variables
  - most general ODE system
  - Assume L linear, but only for convenience

- Let $\hat{u}(t;\epsilon)$ be solution computed by ODE integrator at times

$$t^h \equiv t_n = t_{min}, t_{min} + h, t_{min} + 2h, \ldots, t_{max}$$

- Consider second-order centred FDA of $Lu = 0$

$$L^h u^h = 0$$

$$L^h = L + h^2 E_2 + O(h^4) \tag{9.5}$$

Where E is higher order differential operator (higher than L)

- Note that this equation (34) defines u^h and that

$$u^h(t) \neq \hat{u}(t^h; \epsilon)$$

where the u hat term is ODE, integrator

- Again, L^h can be expanded

$$L^h = L + h^2 E_2 + h^4 E_4 + \dots$$

Where the E terms are higher order differential operators

- Can write

$$\hat{u}(t; \epsilon) = u(t) + e(t; \epsilon)$$

Error in computed solution from ODE integrator

- Consider action of L^h on $\hat{u}(t; \epsilon)$

$$L^h \hat{u}(\epsilon) = (L + h^2 E_2 + h^4 E_4 + \dots)(u + e(t))$$

$$= 0Lu + h^2 E_2 u + \dots L^h e(t)$$

where the last term is small compared to other term

$$\Rightarrow L^h \hat{u} \approx h^2 E_2 u = h^2 r = O(h^2)$$

where r is some function

$$\text{Assuming} \quad h^2 E_2 \gg L^h e(t)$$

- Makeup Monday October 12th 2023
- With high accuracy ODE solver is usually possible to satisfy this equation, at least over some time interval (t_{min}, t_{max}) and as long as h isn't too small.
- KEY idea is to show/check **correctness** of implementation; e.g. checking for errors in coding of equations

9.4.3 Simple Harmonic Motion Example

- Governing D.E. (unit angular frequency; 0 phase)

$$\frac{d^2y(t)}{dt^2} = -y \tag{9.6}$$

- first order form; define

$$y_1(t) \equiv y(t) \qquad y_2(t) \equiv \frac{dy_1}{dt}$$

- Then Equation of motion becomes system

$$\frac{dy_1}{dt} = y_2 \qquad \frac{dy_2}{dt} = -y_1$$

Subject to initial conditions

$$y_1(0) = y(0) \qquad y_2(0) = \frac{dy}{dt}(0)$$

Independant Residual Evaluation

- First rewrite (7) as

$$\frac{d^2y(t)}{dt^2} + y(t) = 0 \tag{9.7}$$

- Now, use ODE integrator to generate solution $\hat{u}(t^h;\epsilon)$ on a level-l uniform mesh

$$t_n^h = 0, h, 2h, \ldots, t_{max}$$

with

$$h = \frac{t_{max}}{2^l}$$

- Then apply $O(h^2)$ FDA of equation (36) to $\hat{y}$ to compute residual R_n

$$R_n \equiv \frac{\hat{y}_{n+1} - 2\hat{y}_n + \hat{y}_{n+1}}{h^2} + \hat{y}_n \qquad n = 2, 3, \ldots, 2^l - 1$$

- Should find that

$$\text{RMS}(R_n) = O(h^2)$$

- In particular, compute R_n on 3 separate levels of discretization

$$h_1 = h_e$$
$$h_2 = 2h_1 = h_{l-1}$$
$$h_3 = 4h_1 = h_{l-2}$$

But using single level-l calculation using the ODE integrator

- Then by Plotting

$$16R^l, \quad 4R^{l-1}, \quad R^{l-1}$$

On a single graph, should see near-coincidence of the curves $\Rightarrow$ convergence of independent residual.

9.5 Passing Additional Arguments to the Derivatives Function

- Systems of ODE often have additional parameters (Change from solution to solution)
- The calling sequence for the derivatives function is fixed

$$\text{function derivative } = \text{odefun}(t, y)$$

- Need mechanism to pass in additional information
- At least 3 ways

9.5.1 Use an anonymous function definition

– Example: 3 Parameter a,b,c

```
code function    odefun(t,y,a,b,c)
a=1.0
b = 2.5
c =3.14
[tout yout] = ode45(@(t,y) odefun(t,y,a,b,c, ... tspan, yphi)
```

9.5.2 Use a Global Variable

– Be default, scope of matlab variable is local to programming unit (Script, function)
– Change this behaviour by using global command
– Must use declaration in all programming units in which we want access to variable

```
% Script; function-caller
    .
    .
    .
global m;
m = 1.56;
    .
    .
    .
[you yout] = ode45(@fcn, [0.0 1.0], [0.0 2.0]);
    .
    .
    .
function derivs = fcn(t,y)
    global m;
        .
        .
```

.

```
end
```

– Syntax for multiple global variables:

```
global a b c;
```

Note: no commas

9.5.3 Adjoin parameter to ODE system

– Treat any parameter as a **dependent** variable with a trivial governing ode

– for example

$$\frac{dm}{dt} = 0 \qquad m(0) = 1.56 \Rightarrow m = \text{const} = 1.56$$

– A bit more computationally intensive than the other two approaches; and a bit more complicated to implement.
$\Rightarrow$ (Matt) Advocates using anonymous functions (although he uses global variables).

- See example of dumbbell orbit slides from the course site

9.6 The one dimensional Toda Lattice

(following https://www.mat.univie.ac.at/ gerald/ftp/book-jac/toda.html)

Model for crystal (simple solution), particle sites with spring between sites:

. . . . . .

with nearest neighbour interactions

- Let g(n,t) be displacement of particle n from its equilibrium position; p(n,t) its momentum (m=1) and V(r) the interaction potential where r is the interparticles separation.
- Hamiltonian

$$\mathcal{H}(p,q) = \sum_n \frac{p(n,t)^2}{2} + V(q(n+1,t) - q(n,t))$$

- Example: Harmonic Interaction

$$V(r) = r^2$$

- Resulting set of equations are linear, with constant set of coefficients
- Solution: super position of normal modes
- **Solitons**: pulse like waves which propagate and interact similarly to particles
- **Nonlinear**: Dispersion balances non linearity (Focusing)

- Motivation in part by soliton theory, TODA (1967) came up with a potential

$$V(r) = e^{-r} + r - 1$$

  where the r-1 term means we get harmonic behaviour in small r limit

- Equations of Motion:

$$\frac{d}{dt}p(n,t) = e^{-(q(n,t)-q(n-1,t))} - e^{-(q(n+1,t)-q(n-1,t))}$$

$$\frac{d}{dt}q(n,t) = p(n,t) \qquad n = 1, 2, \ldots, n_{sites}$$

- Can show that the following is a solution

$$q_1(n,t) = q_+ + \log(\frac{N}{D}$$

$$N = 1 + \frac{\gamma}{1 - e^{-2\kappa}}\exp(-2\kappa n + 2\sigma \sinh(k)t)$$

$$D = N \qquad \text{BUT} \qquad n \to n+1$$

- Can put some example values in like

$$\kappa = \gamma = q_+ = 1.0 \qquad \text{and} \quad \sigma = 1$$

  and we get a ”kink” solution (looks like heaviside function sorta)

- Propogrates to the right is sigma is +1 and left if sigma is -1
- Speed of propagation is $\sinh(\kappa)/\kappa \Rightarrow$
  Thinner kinks travel with higher speeds
- Refer to Slide on Course Site for ODE MATLAB Implementation of this Toda Lattice Problem

October 18th 11:07am
THE VAN DER POL OSCILLATOR

(following Tsatsos ...)

- 1920s Van der Pol (VdP) experimented with oscillations in a vacuum tube triode circuit
- Concluded that all initial conditions converged to some finite-amplitude periodic category
- Proposed a non-linear ODE to model phenomena
- (unforced VdP equation)

$$\frac{d^2x}{dt^2} + a(x^2 - 1)\frac{dx}{dt} + x = 0 \tag{9.8}$$

  Where a is constant (parameter) and when a=0 we get linear oscillator

- Version of equation 37 here that includes forcing term

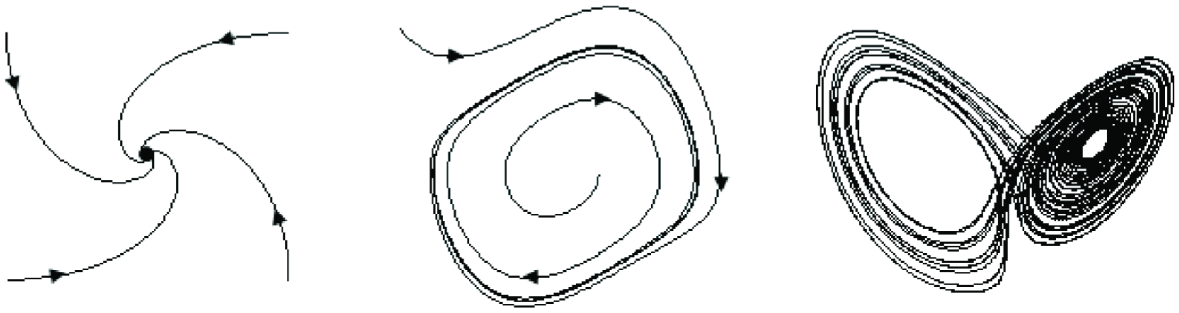
$$\frac{d^2x}{dt^2} + a(x^2 - 1)\frac{dx}{dt} + x = bs\sin(\omega t)$$

  where b and ω are parameters (constants)

- Model has application in: physics, electronics, biology, neurology, sociology, economics
- view VdP equation as "typical" nonlinear dynamical system; illustrates some of qualitative ways we can describe such systems

9.6.1 Analysis Methods

1. **Phase Space Plots**: Good for 2-D systems (one x, one p (dx/dt); plotting trajectories in phase space $\Rightarrow$ reveal nature of dynamics



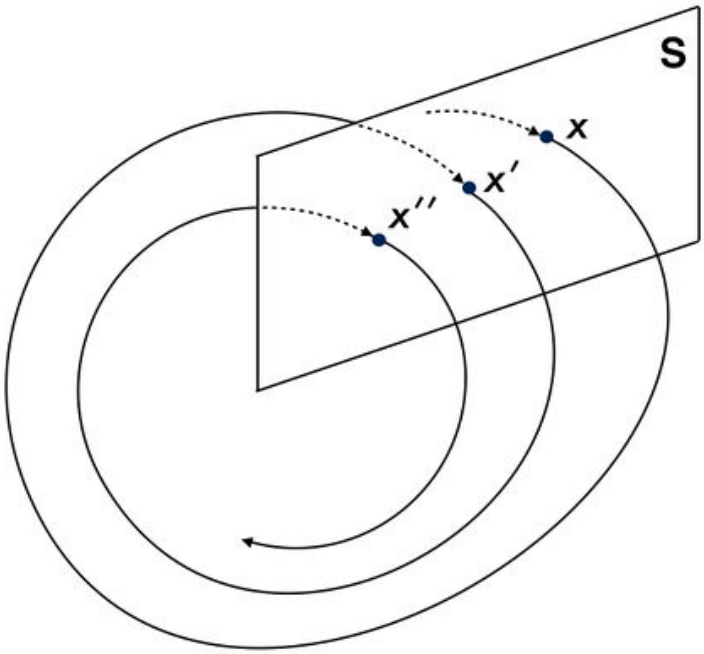
Fixed point attractor | Limit-cycle attractor | Strange attractor

2. Nature of long-time attractors
   (a) **Fixed Point:** trajectory converges to a single point in phase space
   (b) **Limit Cycle**: (Periodic Trajectories) Trajectory converges to closed curve in phase space (undamped oscillator)
   (c) **Strange Attractor**: Trajectory converges to "curve" with fractional dimension in phase space

   Coastline structure on all scales

$$l \approx x^{(d-1)} \qquad 1 \leq d \leq 2$$

3. **Poincare Section:** view dynamics periodically sometimes projecting onto a space with reduced dimensionality (1d on 2d typically) for case of driven dynamics, natural period is period of forcing. Resulting trace of dynamics is called Poincare section. Various features can appear



(a) Fixed Points: single points on Poincare section

(b) Limit Cycles: will appear as one or more isolated points

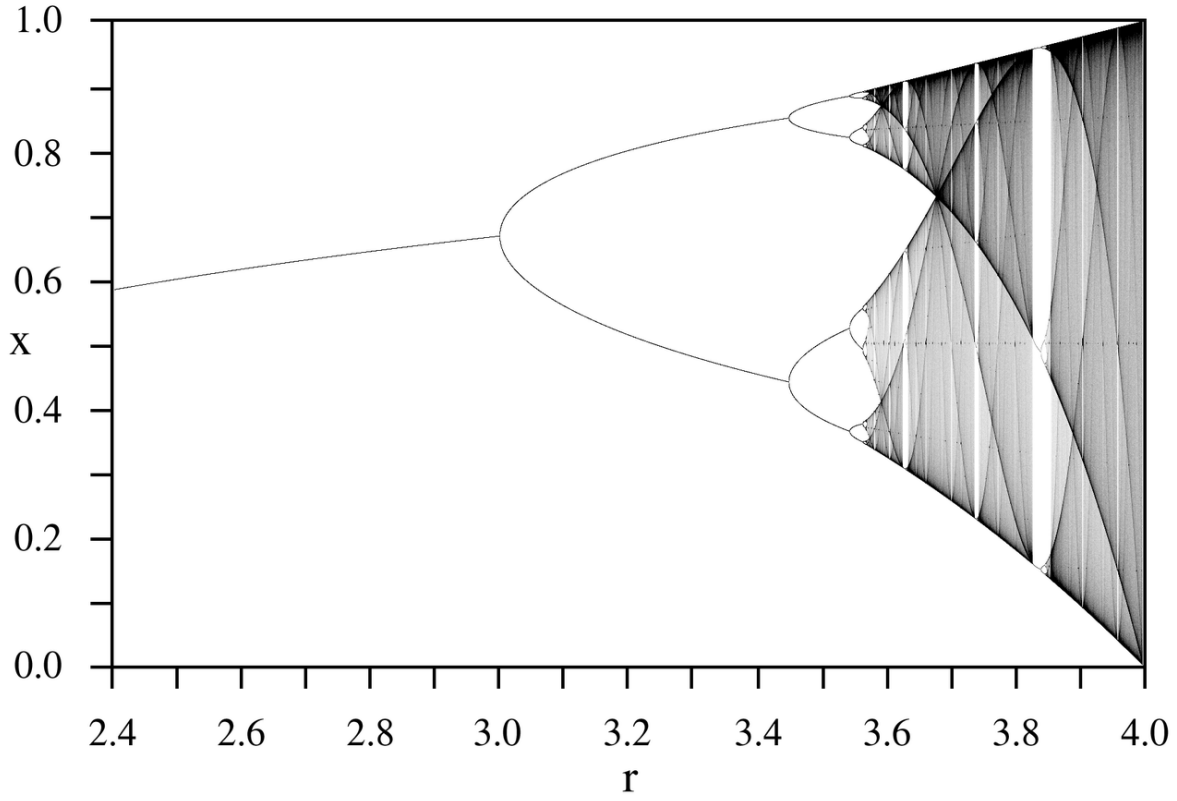
(c) Almost periodic trajectory: will appear as a near-continuous curve

(d) Chaotic dynamics: will appear as a ”curve” with fractional dimensions

4. Bifurcation Diagram

   Plots which show dependence of Poincare section on a problem parameter typically, plot a certain number of points in a given Poincare section, as a function of the parameter

   Typical use here; show transition to chaos (period doubling for example)



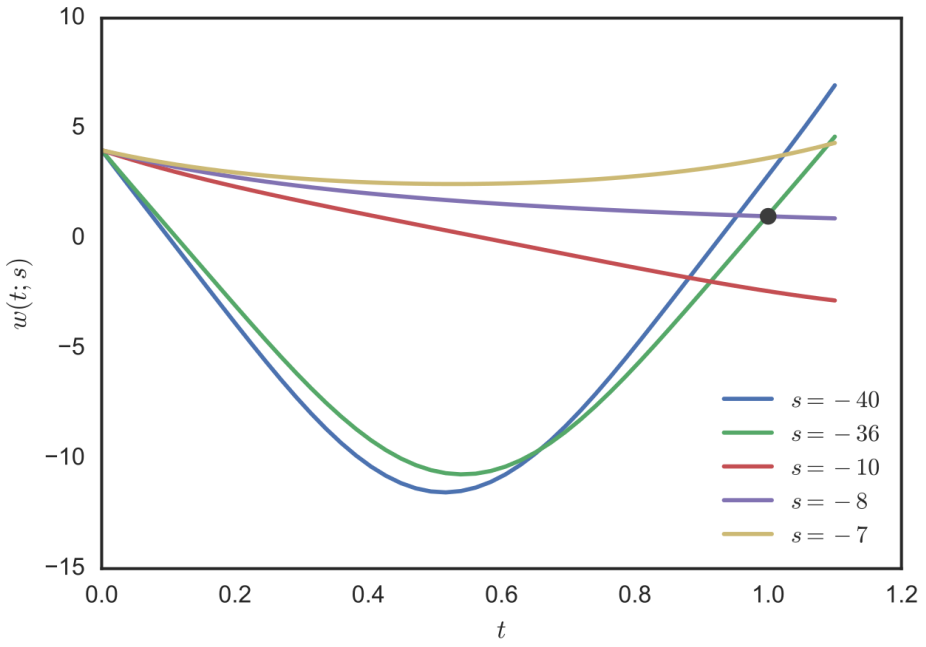
October 20th 2023, 11:00am

9.7 Boundary Value Problems - Shooting

1. Introduction

Recall: 2-PT Boundary value problems; B.C.'s supplied at two points - typically end points of the solution domain

- In idealized physical problems, one of the points will often be $x = \infty$



- * Refer to SOLVING BVPs WITH THE ode45 INTEGRATOR AND SHOOTING* slides

End of the ODE section!

Monday

10 Basic Finite Difference Techniques for Time Dependent PDEs

We can divide time-dependent PDEs into two broad classes:

1. **Initial-value Problems (Cauchy Problems)** spatial domain has no boundaries (either infinite or "closed" – e.g. "periodic boundary conditions"
2. **Initial-Boundary-Value Problems**, spatial domain *finite*, need to specify boundary conditions

Note: Even if *physical* problem is really of Type 1, finite computational resources $\rightarrow$ finite spatial domain $\rightarrow$ approximate as Type 2; will hereafter loosely refer to either type as an IVP.

Working Definition: **Initial Value Problem**

- State of physical system arbitrarily (usually) specified at some initial time $t = t_0$.
- Solution exists for $t \geq t_0$; uniquely determined by equations of motion (EOM) and boundary conditions (BCs).

Issues in Finite Difference (FD) Approximation of IVPs

- Discretization (Derivation of FDA's)
- Solution of algebraic systems resulting from discretization
- Consistency
- Accuracy
- Stability
- Convergence
- Dispersion/ Dissapation
- Treatment of Nonlinearities
- Computational Cost: Expect O(N) work, where N is the total number of spacetime grid points used

10.1 Type of IVP (By example, 1 space fim and 1 time)

- Suppress boundary conditions for time being

10.1.1 Wave and Wave-like ("hyperbolic equations"): The 1-d wave equation

$$u_{tt} = c^2 u_{xx} \qquad u = u(t,x) \quad c \epsilon \mathcal{R} \tag{10.1}$$

with initial conditions:

$$u(x,0) = u_0(x)$$

$$u_t(x,0) = v_0(x)$$

10.1.2 Diffusion ("Parabolic"): The 1-d diffusion equation

$$u_t = \sigma u_{xx} \qquad \sigma \epsilon \mathcal{R}, \quad \sigma > 0 \tag{10.2}$$

with initial conditions:

$$u(x,0) = u_o(x)$$

10.1.3 Schrodinger: The 1-d Schrodinger equation

$$i\psi_t = -\frac{\hbar^2}{2m}\psi_{xx} + V(t,x)\psi \qquad \psi(t,x)\epsilon\mathcal{C} \tag{10.3}$$

$$\psi(x,0) = \psi_0(x)$$

- Although $\psi(t,x)$ is complex in this case, can rewrite equation above as a system of 2 coupled scalar real-valued equations

10.2 Basic Concepts: Definitions (mostly review)

10.2.1 Differential equation (ignoring BC's)

forcing function, possible 0

$$L_{LL} = f$$

where L term is a differential operator

10.2.2 Difference equation (assume characterized by single discretization scale)

F.D. operator here

$$L^h u^h = f^h$$

10.2.3 Residual

- write FDA in canonical form

$$L^h u^h - h^h = 0$$

- $\tilde{u} \Rightarrow$ be some approximation of u^h, then residual

$$r^h \equiv L^h \tilde{u}^h - h^h \qquad \tilde{u}^h \to u^h; \quad r^h \to 0$$

10.2.4 Truncation error (don't confuse with solution error)

- Truncation error τ^h of FDA is defined by

$$\tau \equiv L^h u - f^h$$

  where u is the continuum solution

- Note: u satisfies the continuum D.E.
- Form of T.E. can always be computed (Taylor Series) from the FDA and the D.E.

10.2.5 Convergence

- Assume FDA characterized by single scale h, then u^k converegs iff

$$u^h \to u \quad as \quad h \to 0$$

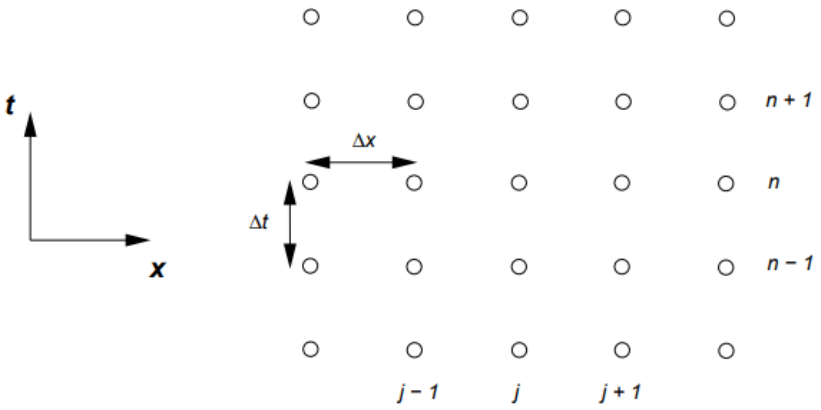


Figure 10.1: Portion of uniform finite-difference mesh (grid) for 1-d time-dependent problem. Note that the spacings in both the spatial and temporal directions are constant

10.2.6 Consistency

- With same assumption, say FDA is consistent iff

$$\tau^h \to 0 \quad as \quad h \to 0$$

- Consistency is a necessary condition for convergence (not sufficient though)

10.2.7 Accuracy

- Assuming that the FDA is characterized by a single discretization scale, h, we say that the FDA is p-th order accurate or simply p-th order if

$$\lim_{h\to 0}\tau^h = O(h^p)$$

10.2.8 Solution Error

- Solution error associated with FDA is

$$e^h \equiv u - u^h$$

10.3 Sample Discretizations (FDAs): 1-d Differential Equation

$$u_t = \sigma u_{xx} \qquad (c = 1) \qquad 0 \le x \le 1; \quad t \ge 0 \tag{10.4}$$

- Step 1: Discretize Domain (uniform grid) around points of form (x_j, t^n) as shown in the figure below

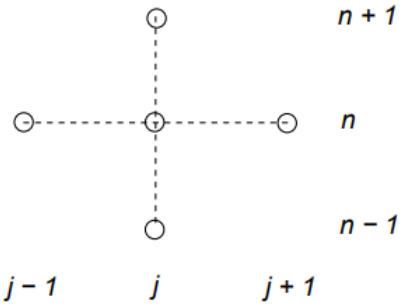


Figure 10.2: Stencil (molecule/star) for "standard" $O(h^2)$) approximation of (15).

$$t^n \equiv n\Delta t, \qquad n = 0, 1, 2, \ldots \tag{10.5}$$
$$x_j \equiv (j-1)\Delta x, \qquad j = 1, 2, \ldots J \tag{10.6}$$
$$u_j^n \equiv u((n-1)\Delta t, (j-1)\Delta x) \tag{10.7}$$
$$\Delta x = (J-1)^{-1} \tag{10.8}$$
$$\Delta t = \lambda \Delta x, \qquad \lambda \equiv \text{Courant number} \tag{10.9}$$

- Step 2: Write down FDA for DE (see Figure 10.2)

Discretized interior equation:

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = (u_{tt})_j^n + \frac{1}{2}\Delta t (u_{tt})_j^n + O(\Delta t^2) \tag{10.10}$$
$$= (u_{tt})_j^n + O(\Delta t) \tag{10.11}$$

$$\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} = (u_{xx})_j^n + \frac{1}{12}\Delta x^2 (u_{xxxx})_j^n + O(\Delta x^4) \tag{10.12}$$
$$= (u_{xx})_j^n + O(\Delta x^2) \tag{10.13}$$

FDA

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \sigma \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \tag{10.14}$$

Where we can view the u_j^{n+1} term as the equation for the unknown.

- If we know u_j^n for all j, then can compute u_j^{n+1} for all j (given B.C.'s)
- Initialize scheme using

$$u_j^1 = u_0(x_j)$$

where u is the given function

- Easy to solve explicitly for $u_j^{n+1} \Rightarrow$ Scheme is called explicit

*** Exam Hint Good question *** :

- Be able to discretize all continuum equations that that we use in class (for FDA)
- Be able to find the truncation error of these steps

Truncation Error

- Interior equation

$$u_t = \sigma u_{xx} = (\partial_t - \sigma \partial_{xx})u = 0$$

- FDA

$$^h_t - \sigma^h_{xx})u^h = 0$$

  Where the partial terms in front of the u expression are the L operator and L^u operator as discussed in the truncation section earlier.

- Truncation Error

$$\begin{aligned} \tau^h &\equiv L^h u \\ &= (\partial_t^h - \sigma \partial_{xx}^h)u \end{aligned}$$

$$\tau^h = (\partial_t^h - \sigma \partial_{xx}^h)u = (\partial_t - \sigma \partial_{xx})u + \dots$$

where the term in from of u equals to $0 \Rightarrow$ consistency

$$\tau^h = \frac{1}{2}\Delta t(u_{tt})_j^n - \frac{1}{12}\sigma \Delta x^2 (u_{xxxx})_j^n + O(\Delta t^2) + O(\Delta x^4)$$

above: Leading order truncation terms

$$\tau^h = O(\Delta t) + O(\delta x^2) = O(\Delta t, \Delta x^2)$$

above: "First order in time; second order in space"
**"" Guranteed to be on final exam "" ** -Matt

Discretize Boundary Conditions (Homogeneous Dirichlet):

$$u_1^{n+1} = u_J^{n+1} = 0$$

Discretize Initial Conditions:

$$u_j^1 = u_0(x_j)$$

Iterative solution procedure
set $u_j^1 \to u_j^2 \to u_j^3 \to \dots$
Need solution not to blow up (will almost certainly be physical)

10.4 Stability Analysis

- Heuristic notion of stability: size of numerical solution does not change "too much" as a function of time:

$$||u(t,x)|| \sim ||u(x,0)|| \tag{10.15}$$

  Where $||\cdot||$ is some suitable norm, such that the L_2 norm

$$N(t) = ||u(t,x)|| \equiv \left(\int_0^1 u(t,x)^2 dx \right)^{1/2} \tag{10.16}$$

- Write time-dependent FDA in form

$$\mathbf{u}^{n+1} = \mathbf{G}[\mathbf{u}^n],$$

  where $\mathbf{G}$ is some update operator (Matrix), and $\mathbf{u}$ is a column vector.

- Can always write FDA in this form by introducing auxiliary variables analogous to what we did for ODEs

- Solution at time step n can be written as

$$\mathbf{u}^n = \mathbf{G}^{(n-1)}\mathbf{u}^0$$

- Now, assume that G matrix has complete set of orthonormal eigenvalues:

$$\mathbf{e}_k, \quad k = 1, 2, \ldots J$$

  and associated eigenvalues

$$\mu_k, \quad k = 1, 2, \ldots, J$$

  so that

$$\mathbf{G}\mathbf{e}_k = \mu_k \mathbf{e}_k \qquad k = 1, 2, \ldots, J$$

- Can now decompose initial data (spectral decomposition)

$$\mathbf{u}^0 = \sum_{k=1}^{J} c_k^0 \mathbf{e}_k$$

- then

$$\mathbf{u}^n = \mathbf{G}^{n-1} \left(\sum_{k=1}^{J} c_k^0 \mathbf{e}_k \right) \tag{10.17}$$

$$= \sum_{k=1}^{J} c_k^0 (\mu_k)^{n+1} \mathbf{e}_k \tag{10.18}$$

  Where we need the mu term to stay bounded. and the mu term could be complex.

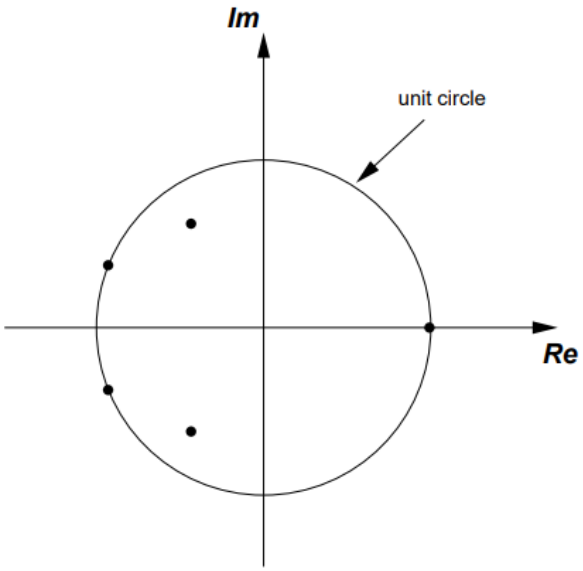


Figure 10.3: Schematic illustration of location in complex plane of eigenvalues of update matrix G. In this case, all eigenvalues (dots) lie on or within the unit circle, indicating that the corresponding finite difference scheme is stable.

- Clearly, if the scheme is to be stable, we must have

$$|\mu_k| \leq 1 \qquad k = 1, 2, \ldots J$$

10.5 Von-Neumann (Fourier) Stability Analysis

- Von-Neumann stability analysis is based on the ideas sketched in the figure, but additionally assumes
  - that the difference equation is linear with constant coefficients,
  - that the boundary conditions are periodic.
- Can use Fourier Analysis: which has the same benefits in the discrete domain – difference operators in real-space variable x $\rightarrow$ algebraic operations in Fourier-space variable k – as it does in the continuum
- Schematically, instead of

$$\mathbf{u}^{n+1}(x) = \mathbf{G}[\mathbf{u}^n(x)],$$

- we consider the Fourier-domain equivalent

$$\tilde{\mathbf{u}}^{n+1}(k) = \tilde{\mathbf{G}}[\tilde{\mathbf{u}}^n(x)],$$

- where k is the wave number (Fourier-space variable). and the tilde denotes the Fourier transformed version of the different terms.

- Define Fourier-transform grid function via

$$\tilde{\mathbf{u}}^n(k) = \frac{1}{\sqrt{2\pi}} \int_{\infty}^{\infty} e^{-ikx} \mathbf{u}^n(x) dx \tag{10.19}$$

- For a general difference scheme, we will find that

$$\tilde{\mathbf{u}}^{n+1}(k) = \tilde{\mathbf{G}}(\xi)\tilde{\mathbf{u}}^n(k)],$$

$$\xi = kh = k\Delta x$$

  Will need to show that $\tilde{\mathbf{G}}(\xi)$'s eigenvalues lie within or on the unit circle for all conceivable ξ.

- if $\tilde{\mathbf{G}}(\xi)$ is a scalar, then its modulus must be ≤ 1

$$\tilde{\mathbf{G}}(\xi) \equiv \text{Amplification Matrix / Factor}$$

- What is appropriate range for $\xi = kh = k\Delta x$?
  - Shortest wavelength representable on uniform mesh with spacing h is $\lambda = 2h$ (Nyquist limit), corresponding to a maximum wave number $k = (2\pi)/\lambda = \pm\pi/h$.
  - Therefore appropriate range is

$$-\pi \leq \xi \leq \pi,$$

- Start with VN stability analysis for diffusion equation from last day
- Introduce undivided difference operator D^2

$$D^2 u(x) = u(x+h) - 2u(x) + u(x-h)$$

- Supress spatial grid index, differential equation is

$$u^{n+1} = u^n + \alpha D^2 u^n$$
$$\alpha = \sigma \frac{\Delta t}{h^2} = \sigma \frac{\Delta t}{\Delta x^2}$$

- What's action of D^2 in fourier space?
- Use inverse F.T.

$$u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ikx} \tilde{k} dk$$

so

$$D^2 u(x) = u(x+h) - 2u(x) + u(x-h) = \int_{-\infty}^{\infty} (e^{ikh} - 2 + e^{-ikh}) e^{ikx} \tilde{u}(k) dk \tag{10.20}$$
$$= \int_{-\infty}^{\infty} (e^{i\xi} - 2 + e^{-i\xi}) e^{ikx} \tilde{u}(k) dk \tag{10.21}$$

- We can replace the term in brackets with $-4\sin^2(\xi/2)$:

$$D^2u(x) = u(x+h) - 2u(x) + u(x-h) = \int_{-\infty}^{\infty} \left(-4\sin^2(\xi/2)\right) e^{ikx}\tilde{u}(k)dk$$

- Missed some points
- Amplificaiton Factor in Fourier space:

$$\tilde{\mathbf{G}} = 1 - 4\alpha\sin^2(\xi/2) \tag{10.22}$$

- Thus, for stability must have

$$4\alpha\sin^2(5/2) \leq 2 \quad \rightarrow \quad \alpha \leq \frac{1}{2}$$

$$\alpha = \sigma\frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \tag{10.23}$$

- But of bad news: as $\Delta x \to 0, \Delta t$ must $\to 0$ quadratically in Δx
- In following, will drop integrals, $\frac{1}{\sqrt{2\pi}}$, e^{ikx} and simply write down amplification matrix when it is clear what it is

10.6 Some other discretizations of the diffusion equation

10.6.1 Implicit First-Order

- One general way to improve stability of FDAs is to use implicit schemes
- Use same mesh, different stencil
- FDA

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \sigma\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}$$

- Perform T.S. expansions about (x_j, t^{n+1}) get T.E. (verify!)

$$\tau = -\frac{1}{2}\Delta t(u_{tt})_j^{n+1} + \frac{1}{12}\sigma\Delta x^2(u_{xxxx})_j^{n+1} + O(\Delta t^2) + O(\Delta x^2)$$

$$\tau = O(\Delta t, \Delta x^2)$$

First order in time, second order in space

- Can't explicitly write down formula for u_j^{n+1} due to coupling of unknowns at $t = t^{n+1}$
- Scheme is implicit
- Have to solve (linear) system of equations for $u_j^{n+1}, \qquad j = 1, 2, \ldots, J$
- Defer solution discussion until stability properties derived

- Write scheme as

$$u_j^{n+1} = u_j^n + \alpha D^2 u_j^{n+1}$$

where we are using α, D as previously mentioned

$$\Rightarrow (1 - \alpha D^2) u^{n+1} - u^n$$

- Now apply F.T., get (verify!)

$$(1 + 4\alpha \sin^2 \frac{\xi}{2} \tilde{u}(k)^{n+1} = \tilde{u}(k)^n$$

- So, amplification factor is

$$\tilde{G}(\xi) = \frac{1}{1 + 4\alpha \sin^2 \frac{\xi}{2}}$$

$$\Rightarrow \tilde{G}(\xi) \leq 1 \text{ For ALL } \alpha, \xi$$

Such a scheme is said to be unconditionally stable (in the VN sense)

- VN stability necessary but not necessarily sufficient for "True" stability.

October 30th 2023

- Set $\sigma = 1$ and rearrange earlier equation:

$$(-\Delta x^{-2}) u_{j+1}^{n+1} + (\Delta t^{-1} + 2\Delta x^{-2}) u_j^{n+1} + (-\Delta x^{-2}) u_{j-1}^{n+1} = (\Delta t^{-1}) u_j^n, \qquad j = 2, \ldots, J-1$$

$$u_1^{n+1} = u_J^{n+1} = 0 \qquad \text{(Homogeneous, Dirichlet. B.C.;}$$

- In matrix form

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{b}$$

What is the structure of matrix A? (non-zero)

  - Tri-diagonal system. i.e. three diagonals but everything else is 0.

- Special case of **sparse** matrix, defined as a matrix with a majority of 0 elements (roughly 50 percent or more)
- Here, for large J, vast majority of elements will be 0.
- Can optimize solution of linear system W.R.T. the sparsity structure
- For tridiagonal systems can compute solution in $O(J)$ time vs general $O(J^3)$, BUT have to take advantage of sparsity structure
- Recommended approach in MATLAB:

```
spdiags  % (sparse diagonals)
```

- Illustrate using example which solves above system ($nx = J$)
- Define diagonals of sparse matrix **A** as columns of auxiliary matrix **B**, pass **B** along with diagonal numbers to spdiags

```
% Initialize storage for sparse matrix and RHS

dl = zeros(nx, 1);
d = zeros(nx,1);
du = zeros(nx,1);
f = zeros(nx,1);

% Set up tridiagonal system

dl = -1.0/dx^2 * ones(nx,1);
d = (1.0/dt * 2.0/dx^2) * ones(nx,1);
du = dl;

% Fix up boundary cases

% Left B.C.
d(1) = 1.0;
du(2) = 0.0;

% Right B.C.
dl(nx-1) = 0.0;
d(nx) = 1.0;

% Define sparse matrix
% B is [dl d du]
% -1:1 is which diagonals
% nx, nx is dimensions

A = spdiags([dl d du], -1:1, nx, nx)

% Define RHS vector

f(2:nx-1) = u(n, 2:nx-1)/dt;
f(1) = 0;
f(nx) = 0;
```

- Can then use **A** in linear solve via left division

```
u(n+1,:) = A\f;

% \ operator knows about sparse matrices; solve is optimized, very fast!
```

- Can see full matrix (with all 0's included using the full command:

```
full(A)
```

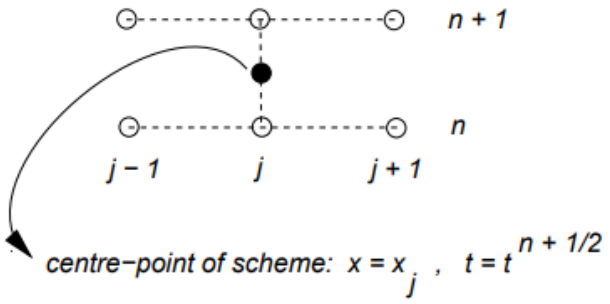


Figure 10.4: : Stencil (molecule/star) for $O(h^2)$ Crank-Nicholson approximation of (19)

- Diagonal in position M from main diagonal has length nx-m
- Thus, any column of **B** other than the main diagonal will have positions/values that are not used
- Diagonal above (super diagonal): At beginning
- Diagonal below (sub diagonal): At end
- Once more, note how B.C's are implemented

$$(1)u_1^{n+1} + (0)u_2^{n+1} = 0$$

$$(0)u_{J-1}^{n+1} + (1)u_J^{n+1} = 0$$

10.6.2 Crank-Nicholson Scheme

$$u_t = \sigma u_{xx}$$

Using rectangle stencil with center fictitious point at n+1/2 and j. And that is where you do the taylor series about.

FDA

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = \frac{1}{2}\sigma\left[\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2} + \frac{u_{j+1}^{n} - 2u_j^{n} + u_{j-1}^{n}}{\Delta x^2}\right] \tag{10.24}$$

November 1st 2023

Truncation error takes quite a bit of work; here is the result:

$$\tau = \frac{1}{24}\Delta t^2 (u_{ttt})_j^{n+\frac{1}{2}} - \frac{1}{8}\sigma\Delta t^2 (u_{ttxx})_j^{n+\frac{1}{2}} - \frac{1}{12}\sigma\Delta x^2 (u_{xxxx})_j^{n+\frac{1}{2}} + O(\Delta t^4) + O(\Delta x^4) + O(\Delta t^2\Delta x^2)$$

$$\tau = O(\Delta t^2, \Delta x^2) \Rightarrow \text{ Improved temporal accuracies relative to first two schemes}$$

- "Wouldnt ask to derive a truncation error for a scheme like this on the exam but he WOULD ask for us to derive truncation error for previous two schemes"
- Make sure you can derive tau for simpler (first order) schemes, ** Exam Hint **

Stability

- Write scheme
$$(1 - \frac{\alpha}{2}D^2)u_j^{n+1} = (1 + \frac{\alpha}{2}D^2)u_j^n$$
α, D^2 as previously used
- Apply Fourier Transform (get $\rightarrow$ verify)
$$(1 + 2\alpha \sin^2 \xi/2)\tilde{u}(k)^{n+1} = (1 - 2\alpha \sin^2 \xi/2)\tilde{u}(k)^n$$
- Amplification factor is
$$\tilde{G}(\xi) = \frac{1 - 2\alpha \sin^2 \xi/2}{1 + 2\alpha \sin^2 \xi/2}$$
- This is of the form
$$\tilde{G}(\xi) = \frac{1 - x}{1 + x} \qquad x \geq 0$$
$$\tilde{G}(\xi) \leq 1 \text{ for all } \xi, \alpha$$
Scheme is also **unconditionally** stable
- Solution with Crank method yields same structure as the t grid
$\Rightarrow$ tridiagonal system
- Solve using spdiags as we did for implicit scheme (tutorial next week)

10.6.3 Exact Solution for Testing

- Recall
$$u_t = \sigma u_{xx}$$
on
$$0 \leq x \leq 1 \qquad t \geq 0$$
$$u(x, 0) = u_0(x)$$
$$u(0, t) = u(1, t) = 0$$
- Exact solution
$$u(x, t) = e^{-\sigma\omega^2 t} \sin(\omega t)$$
where $\omega = n\pi, \qquad n = 1, 2, \ldots$
$$u_t = u_{xx} = -\sigma\omega^2 u(t, x)$$

10.7 The 1-D Schrödinger Equation

$$i\psi_t = -\frac{\hbar^2}{2m}\psi_{xx} + V(t,x)\psi \tag{10.25}$$

$$0 \leq x \leq 1$$

$$\psi(t,x) \in \mathbb{C}$$

V(t,x) is given potential

- Non-dimensionalize, solve on unit interval with homogeneous Dirichlet B.C.'s

$$i\psi_t = -\psi_{xx} + V\psi$$

$$\psi(0,x) = \psi_0(x)$$

$$\psi(t,0) = \psi(t,1) = 0 \Rightarrow \text{Physical interpretation?}$$

Infinite potential walls at x=0, 1; no penetration of wave function

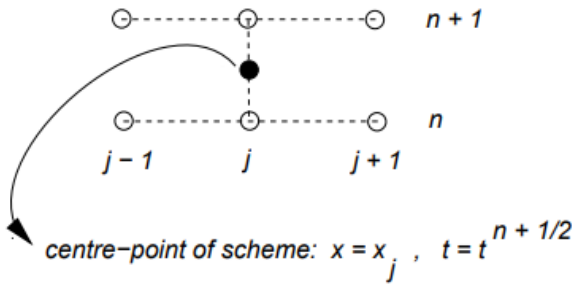
- Exact solution for testing (V=0)

$$\psi(t,x) = e^{i\omega^2 t}\sin(\omega x)$$

where $\omega = n\pi, \quad n = 1,2,\ldots$

FDA:

Use crank-nicholson grid



$$\psi_j^{n+\frac{1}{2}} = \frac{1}{2}(\psi_j^{n+1} + \psi_j^n)$$

with error of $O(\Delta t^2)$

$$i\frac{\psi_j^{n+1} - \psi_j^n}{\Delta t} = -\frac{1}{2}\left(\frac{\psi_{j+1}^{n+1} - 2\psi_j^{n+1} + \psi_{j-1}^{n+1}}{\Delta x^2} + \frac{\psi_{j+1}^n - 2\psi_j^n + \psi_{j-1}^n}{\Delta x^2}\right) + \frac{1}{2}V_j^{n+\frac{1}{2}}(\psi_j^{n+1} + \psi_j^n) \tag{10.26}$$

$$\psi_j^{n+1} = \psi_J^{n+1} = 0$$

- Truncation error

$$\tau = O(\Delta t^2, \Delta x^2)$$

  Second order in space and time

- Stability analysis
- **Theorem**: In stability analysis, can neglect terms that do not include spatial derivatives!
- EXAM HINT FOR 830AM final: "undifferentiated terms you don't need to worry about... etc for stability analysis"
- Write scheme, without $V\psi$ term, as

$$(i + \frac{1}{2}\alpha D^2)\psi^{n+1} = (i - \frac{1}{2}\alpha D^2)\psi^n$$

  D^2 as before

  $\alpha = \frac{\Delta t}{\Delta x^2}$

- Under F.T.

$$(i - 2\alpha \sin^2 \xi/2)\tilde{\psi}(k)^{n+1} = (i + 2\alpha \sin^2 \xi/2)\tilde{\psi}(k)^n$$

- Read off amplification factor

$$\tilde{G}(\xi) = \frac{i + 2\alpha \sin^2 \xi/2}{i - 2\alpha \sin^2 \xi/2}$$

  Which is of the form

$$\frac{1+a}{1-a} \qquad a \text{ REAL}$$
$$\left|\frac{i+a}{i-a}\right| = \frac{|i+a|}{|i-a|} = \frac{1+a^2}{1+a^2} = 1$$

  Thus,

$$|\tilde{G}(\xi)| = 1 \quad \text{for all } \xi, \alpha$$

  The scheme is unconditionally stable

  This reflects fact that probability is conserved in the continuum

  "we will see this in play with final project"

  Solution of equations

- Rewrite equation as tridiagonal system

$$C_j^+ \psi_{j+1}^{n+1} + C_j^0 \psi_j^{n+1} + c_j^- \psi_{j-1}^{n+1} = f_j$$

  where only second term of c has actual j dependence from the V psi term

- Set up matrix using spdiags, solve using left division (project)

10.8 Wave-equation

10.8.1 1-D Wave equation with fixed (dirichlet) B.C.'s

$$u_{tt} = u_{xx} \qquad (c = 1) \qquad 0 \leq x \leq 1 \quad t \geq 0$$

$$u(0, x) = u_0(x)$$

$$u_t(0, x) = u_0(x)$$

$$u(t, 0) = u(t, 1) = 0$$

- Step 1: Discretize domain (use same mesh)
- Make FDA Characterized by single scale, h

$$\Delta t = \lambda \Delta x = \lambda h$$

$$h = \Delta x$$

$$\lambda = \frac{\Delta t}{\Delta x}$$

$$\lambda \equiv \text{"Courant Number"}$$

- Whenever we vary Δx, automatically vary Δt (λ held constant)

- Step 2: Write down FDA:
  MAKE SURE YOU CAN DERIVE THIS FOR EXAM HINT - if you know the left hand side, also know how to get the expansion on the right hand side***

$$(\Delta t)^{-2}(u_j^{n+1} - 2u_j^n + u_j^{n-1}) = (u_{tt})_j^n + \frac{1}{12}\Delta t^2 (u_{tttt})_j^n + O(\Delta t^4)$$

$$= (u_{tt})_j^n + O(\Delta t^2)$$

  hint: Same expansion (can use similarity argument on exam and not repeat full derivation):

$$(\Delta x)^{-2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n) = (u_{xx})_j^n + \frac{1}{12}\Delta x^2 (u_{xxxx})_j^n + O(\Delta x^4)$$

$$= (u_{xx})_j^n + O(\Delta x^2)$$

- FDA

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\Delta t^2} = \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2} \qquad j = 2, 3, \ldots, J-1$$

- Called a "three level scheme" since it couples unknowns on 3 time levels
- Truncation error ($\tau = \tau^h = L^h u$; $L^h u^h = 0$)
  First write FDA in form

$$\tau = \frac{1}{12}\Delta t^2 (u_{tttt})_j^n - \frac{1}{12}\Delta x^2 (u_{xxxx})_j^n + O(\Delta t^4) + O(\Delta x^4)$$

$$= O(\Delta x^2, \Delta t^2) = O(h^2)$$

  Second order in space and time

- Discrete B.C.'s

$$u_1^{n+1} = u_J^{n+1} = 0$$

  Missed info

- Discrete I.C.'s
  - Need to give two "time levels" of data (effectively $u(0,x), u_t(0,x)$

$$u_j^1 \quad , \quad j = 1, 2, \ldots, J$$

$$u_j^2 \quad , \quad j = 1, 2, \ldots, J$$

- Can solve FDA equation above for u_j^{n+1} explicitly

$$u_j^{n+1} = 2u_j^n - u_j^{n-1} + \lambda^2 (u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

- Initialization revisited
- First time level: $u_0(x)$

$$u_j^1 = u_0(x_j)$$

  Second time level: need to determine u_j^2 up to and including $O(\Delta t^2)$ terms

$$u_j^2 = u_j^1 + \Delta t (u_t)_j^1 + \frac{1}{2}\Delta t^2 (u_{tt})_j^1 + O(\Delta t^3) \tag{10.27}$$

$$= u_j^1 + \Delta t (u_t)_j^1 + \frac{1}{2}\Delta t^2 (u_{xx})_j^1 + O(\Delta t^3) \tag{10.28}$$

$$= u_0(x_j) + \Delta t v_0(x_j) + \frac{1}{2}\Delta t^2 u_0''(x_j) \tag{10.29}$$

Stability Analysis (more involved due to 3 time levels)

- Rewrite FDA in "first order" form by introducing auxiliary variables

$$v_j^n = u_j^{n-1} \Rightarrow v_j^{n+1} = u_j^n$$

$$\Rightarrow u_j^{n+1} = 2u_j^n - v_j^n + \lambda^2 (u_{j+1}^n - 2u_j^n + u_{j-1}^n)$$

$$v_j^{n+1} = u_j^n$$

- In matrix form

$$\begin{bmatrix} u \\ v \end{bmatrix}^{n+1} = \begin{bmatrix} 2 + \lambda^2 D^2 - 1 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}^n$$

  Where the D terms is same as D^2 previously

- Under Fourier Transormation, becomes

$$\begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}^{n+1} = \begin{bmatrix} 2 - 4\lambda^2 \sin^2 \xi/2 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \tilde{u} \\ \tilde{v} \end{bmatrix}^n$$

  where the matrix is $\tilde{G}(\xi)$

- Need to determine conditions so $\tilde{G}(I)$ has eigenvalues on or within unit circle
- Characteristic equation (whose roots are the e.v.'s)

$$\begin{vmatrix} 2 - 4\lambda^2 \sin^2 \xi/20 - \mu & -1 \\ 1 & \mu \end{vmatrix} = 0$$

  or

$$\mu^2 + (4\lambda^2 \sin^2 \xi/2 - 2)\mu + 1 = 0$$

  Equation has roots at

$$\mu(\xi) = (1 - 2\lambda^2 \sin^2 \xi/2) \pm ((1 - 2\lambda^2 \sin^2 \xi/2)^2 - 1)^{1/2}$$

  Need sufficient conditions for

$$|\mu(\xi)| \leq 1$$

Equivalently

$$|\mu(\xi)|^2 \leq 1$$

$$\mu(\xi) = (1-Q) \pm ((1-Q)^2 - 1)^{1/2}$$

$$Q \equiv 2\lambda^2 \sin^2 \xi/2$$

- 3 cases to consider

  1. $$(1-Q)^2 - 1 = 0$$
  2. $$(1-Q^2) - 1 < 0$$
  3. $$(1-Q^2) - 1 > 0$$

  Case 1: $Q = 0$ or $Q = 2$, In both cases $|\mu(\xi)| = 1$
  Case 2: $((1-Q)^2 - 1)^{1/2}$ purely imaginary

$$\mu(\xi) = (1-Q)^2 + (1 - (1-Q)^2) = 1$$

  Case 3: $(1-Q)^2 - 1 > 0 \rightarrow (1-Q)^2 > 1 \rightarrow Q > 2$

$$1 - Q - ((1-Q)^2 - 1) < -1$$

  Instability at

$$|\mu(\xi)| > 1$$

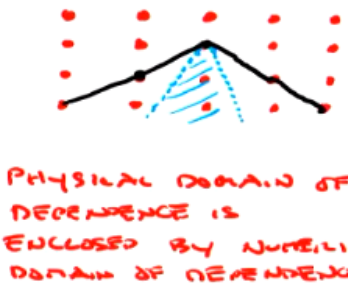
- Necessary condition for von Neumann stability

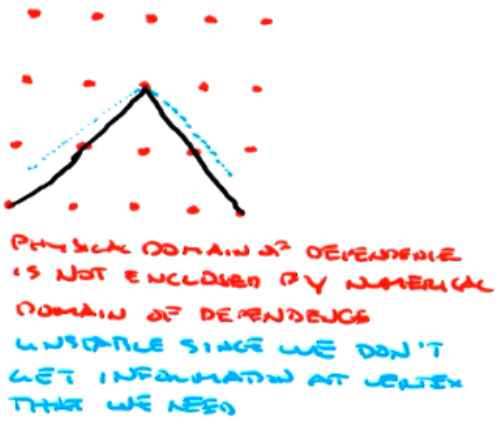
$$(1-Q)^2 - 1 \leq 0 \rightarrow (1-Q)^2 \leq 1 \rightarrow Q \leq 2$$

- But $Q = 2\lambda^2 \sin^2 \xi/2 \quad ; \quad 2\lambda^2 \leq 2; \quad \lambda^2 \leq 1$
  For stability we need

$$\lambda \equiv \frac{\Delta t}{\Delta x} \leq 1$$

- This condition is often called the CFL condition—after Courant, Friedrichs and Lewy who derived it in 1928 (the ratio $\lambda = \Delta x/\Delta t$ is also frequently called the *Courant number*). In practical terms, we must limit the time-discretization scale , Δt, to values no larger than the space-discretization scale, Δx. Furthermore, this type of instability has a "physical" interpretation, often summarized by the statement *the numerical domain of dependence of an explicit difference scheme must contain the physical domain of dependence*.





- EXAM HINT : make sure you can do a stability analysis like this!! Really good exam question

November 6th 11:19am

10.9 Dispersion in FDAs

- Advection Equation

$$u_t = a u_x$$

- Exact solution

$$u(t,x) = f(at + x)$$

  Propogates initial data profile to the left with speed a. No change in profile's shape $\Rightarrow$ non-dispersive. Every fourier component travels with the same speed c.

- Semi-discretization: only discretize in space

$$u_t = aD_x u = a\frac{u_{j+1} - u_{j-1}}{2\Delta x}$$

  Leave time continuous

- Look for normal mode solutions, of the form

$$u = e^{ik(x+G't)}$$

  $k \equiv$ to the wave number
  $a' \equiv$ to the discrete phase speed associated with k

- Substitute in (a')

$$ika'u = \frac{a(2i\sin(k\Delta))}{2\Delta x}u$$

- Solve for a′

$$a' = a\frac{\sin(k\Delta x)}{k\Delta x} = a\frac{\sin\xi}{\xi}$$

$$\xi \equiv k\Delta x$$

- Low frequency limit $\xi \to 0$

$$a' = \lim_{\xi\to 0} a\frac{\sin\xi}{\xi} = a$$

  Expected result

- High frequency limit, $\xi \to \pi$

$$a' = a\frac{\sin\pi}{\pi} = 0$$

  Highest frequency components don't propagate at all

- Typically of low-order FDAs
- In practice may want to damp high-frequency components by adding explicit dissipation to the scheme
- High-frequency components often responsible for instabilities

End of discussion about PDEs in 1-D

11 Solution of problems in two space dimensions

Time-dependent equation that we study, diffusion equation, will serve as a model for the solution of the Schroedinger Equation that we will solve in the Project 2.

11.1 Diffusion Equation

- Model Equation

$$y(x,y,t)_t = \nabla^2 u = u_{xx} + u_{yy}$$

$$u_t = u_{xx} + u_{yy} \qquad \sigma = 1$$

where sigma is the diffusion constant.

On

$$0 \le x \le 1, \quad 0 \le y \le 1; \quad t \ge 0$$

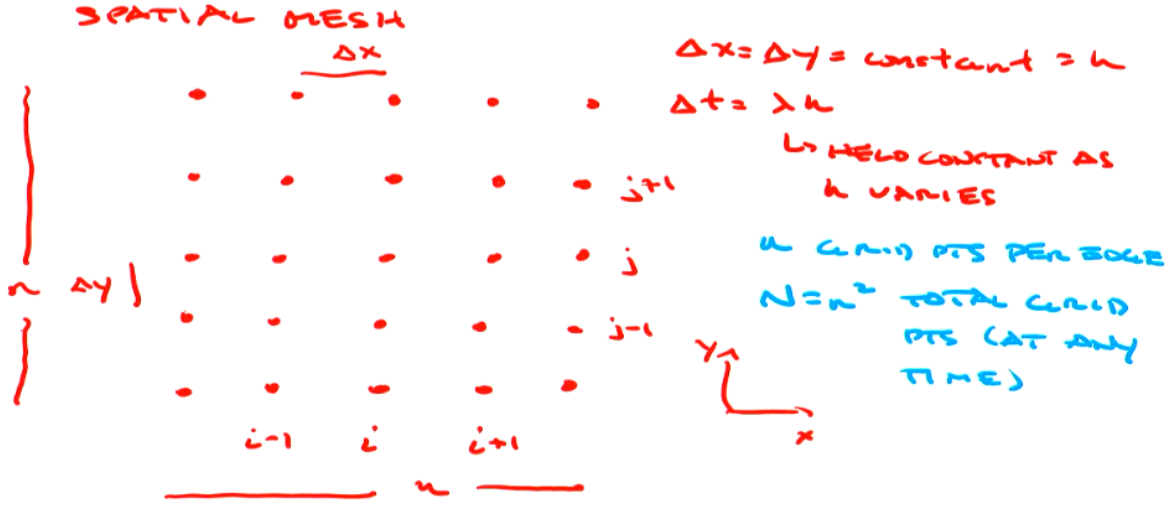
Subject to

$$u(x,y,0) = u_0(x,y)$$

$$u(0,y,t) = u(1,y,t) = u(x,0,t) = u(x,1,t) = 0$$

- Solve by FDA
- Step 1: Discretize Domain

Spatial Mesh



$$\begin{bmatrix} u_{1,1} \\ \vdots \\ u_{n,1} \\ \vdots \\ u_{1,n} \\ \vdots \\ u_{n,n} \end{bmatrix} = \mathbf{u}^n \qquad n^2 = N \text{ components} \tag{11.1}$$

- Grid Function notation

$$u_{i,j}^n = u(x_i, y_j, t^n)$$

11.2 The Crank-Nicholson Scheme For 2D Diffusion Equation

- Assume retains 1-D properties: second-order accuracy, unconditional stability
- Introduce F.D. operators: ${}^h_{xx}$, and ${}^h_{yy}$

$${}^h_{xx}u_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2}$$

$${}^h_{yy}u_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta y^2}$$

- Then scheme is

$$\frac{u_{i,j}^{n+1} - u_{i,j}^n}{\Delta t} = \frac{1}{2}({}^h_{xx} + {}^h_{yy})(u_{i,j}^{n+1} + u_{i,j}^n)$$

  Centre point of scheme

$$(x_i, y_j, t^{n+\frac{1}{2}})$$

- Truncation error ($\Delta x = \Delta y$)

$$\tau = \frac{1}{24}\Delta t^2 (u_{ttt})_{i,j}^{n+\frac{1}{2}} - \frac{1}{8}\Delta t^2((u_{ttxx})_{i,j}^{n+\frac{1}{2}} + (u_{ttxx})_{i,j}^{n+\frac{1}{2}}) - \frac{1}{12}\Delta x^2((u_{xxxx})_{i,j}^{n+\frac{1}{2}} + (u_{yyyy})_{i,j}^{n+\frac{1}{2}}) + O(h^4)$$

$$= O(\Delta t^2, \Delta x^2) = O(h^2)$$

  Second order in space and time

- Define

$$L^h \equiv {}^h_{xx} + {}^h_{yy}$$

  The scheme can be written as

$$(1 - \frac{\Delta t}{2} L^h) u_{i,j}^{n+1} = (1 + \frac{\Delta t}{2} L^h) u_{i,j}^{n}$$

Represents linear system of equations for $u_{i,j}^{n+1}$

$$\mathbf{A}\mathbf{u}^{n+1} = \mathbf{b}$$

Where the u vector is all unknown $u_{i,j}^{n+1}$ arranged row by row E.G.

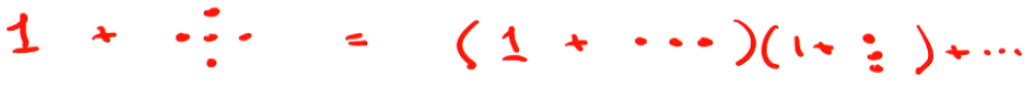
- **A** will again be sparse; what is its sparsity structure?

Think coupling with nearest neighbour left and right ± 1 and then also above and below $\pm n$ (unlikely to ask this on the final exam)

- What is cost of solving systems ? (Recall in 1-D we had tridiagonal system, cost was $O(n) = O(N) \Rightarrow$ Optimal); here $O(n^2) = O(N)$ is optimal
- Bandwidth, W, of matrix is total number of diagonals that span non-zeros portion
- For constant bandwidth NxN systems can be solved in $O(c_w N)$ time when c_w is a constant that depends on c_w; c_w is quadratic in w

November 8th 2023, 11am

- $c_w \sim n^2$ so the order is still $O(N^2)$
- $\Rightarrow$ Direct solution of the linear system even taking into account sparsity structure is too expensive
- IDEA: operator on left is schematically



11.3 Altenating direction implicit method for the diffusion equation

Hint for Project 2, try this before trying for Project 2

Factor operator on left into 1-D operators ⇒ resulting linear systems are tridiagonal and can be solved efficiently.

$$(1 - \frac{\Delta t}{2} L^h) u_{i,j}^{n+1} = (1 + \frac{\Delta t}{2} L^h) u_{i,j}^n$$

- Recall: $L^h = {}^h_{xx} + {}^h_{yy}$, so factor

$$(1 - \frac{\Delta t}{2}{}^h_{xx})(1 - \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^{n+1} = (1 + \frac{\Delta t}{2}{}^h_{xx})(1 + \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^n + \frac{\Delta t^2}{4}{}^h_{xx}{}^h_{yy}(u_{i,j}^{n+1} - u_{i,j}^n)$$

  where the last added term are the quadratic terms from factorization ($O(\Delta t^3)$

$$u_{i,j}^{n+1} = u_{i,j}^n + \Delta t (u_t)_{i,j}^2 + O(\Delta t^2)$$

$$u_{i,j}^{n+1} - u_{i,j}^n = O(\Delta t)$$

  So we have

$$(\quad)(\quad) u_{i,j}^{n+1} = (\quad)(\quad) u_{i,j}^n + O(\Delta t^2)$$

  This defines the one-step update ⇒ for $O(\Delta t^2)$ global error can neglect $O(\Delta t^2)$ term
  Gives one for of ADI scheme

$$(1 - \frac{\Delta t}{2}{}^h_{xx})(1 - \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^{n+1} = (1 + \frac{\Delta t}{2}{}^h_{xx})(1 + \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^n \tag{11.2}$$

- Solve in stages: introduce intermediate grid function $u_{i,j}^{n+\frac{1}{2}}$
  let;

$$(1 - \frac{\Delta t}{2}{}^h_{xx}) u_{i,j}^{n+\frac{1}{2}} = (1 + \frac{\Delta t}{2}{}^h_{xx})(1 + \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^n \tag{11.3}$$

$$(1 - \frac{\Delta t}{2}{}^h_{yy}) u_{i,j}^{n+1} = u_{i,j}^{n+\frac{1}{2}} \tag{11.4}$$

- Key Observation: Each stage requires solution of ñ tridiagonal systems each of which costs $O(n) \Rightarrow$ total cost $= O(n^2) - O(N) \Rightarrow$ optimal
- Specifically
  - Stage 1: for each $j = 2, 3, \ldots, n-1$ solve tridiagonal systems for $u_{i,j}^{n+\frac{1}{2}}$ from $i = 1, 2, \ldots, n$
  - Stage 2: for each $i = 2, 3, \ldots, n-1$ solve tridiagonal systems for $u_{i,j}^{n+1}$ from $j = 1, 2, \ldots, n$
    Loops run over $2, 3, \ldots, n-1$ since B.C.'s fixed, $u_{i,j}^{n+1}$, does not change in time

- Classical version of the ADI method
  - Use fact that $(1 + \Delta t^h_{xx}/2)^{-1}$ and $(1 - \Delta t^h_{xx}/2)$ commute to show that the following form is equivalent

$$(1 - \frac{\Delta t}{2}{}^h{}_{xx})u^{n+\frac{1}{2}}_{i,j} = (1 + \frac{\Delta t}{2}{}^h{}_{yy})u^n_{i,j}$$

$$(1 - \frac{\Delta t}{2}{}^h{}_{yy})u^{n+1}_{i,j} = (1 + \frac{\Delta t}{2}{}^h{}_{xx})u^{n+\frac{1}{2}}_{i,j}$$

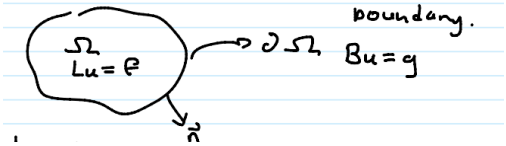
    Again solved in two stages, each involving tridiagonal solves

November 10th 2023 - following Kenneth notes

11.4 Time-Independent: Solving Elliptic PDEs

(Elliptic meaning there is no time dependence)

- Like 2-point BVPs, but in higher dimension. Let's just look at 2D.



Solution domain: Ω

Boundary: $\partial\Omega$

Look for $u(x,y)$ satisfying
$Lu(x,y) = f(x,y)$ on Ω
L – elliptic differential operator
f – source/forcing function
$Bu(x,y) = g(x,y)$ on $\partial\Omega$
B – another operator that encodes BCs
g – another source function

3 Types of BC's

1. Dirichlet - value of u on ∂R is given
2. Neumann - normal derivative of v on $\partial\Omega$ is given.
3. Mixed/Robin - some linear combo of $\alpha(x,y)u + \beta(x,y)\frac{\partial u}{\partial x}$ is given.

11.5 Model Problem

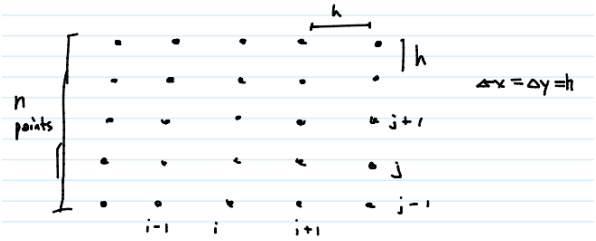
- Poisson equation on unit square with homogeneous Dirichlet Conditions

$$\nabla^2 u(x,y) = u_{xx} + u_{yy} = f(x,y)$$

Subject to

$$u(0,y) = u(1,y) = u(x,0) = u(x,1) = 0$$

2-1 Discretize Model Problem



- **2nd step:** - FDA

$$l^h u^h : f^h$$

$$B^h u^h :?$$

$$u_{xx} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2)$$

$$u_{yy} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2)$$

Substitute into poisson equation

$$\frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4ui,j}{h^2} = f_{i,j} \qquad 2 \leq i,j \leq n-1$$

where n is the number of points one side

Discrete BCs

$$u_{1,j} = u_{n,j} = u_{i,1} = u_{i,n} = 0$$

using the 'plus' stencil

note that the corner points (0,0), (n,0), (0,n), (n,n) are completely separate due to our stencil choice. But it may be more convenient to just incorporate the points.

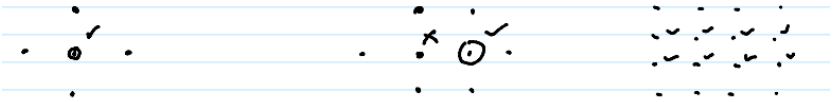
- **3rd Step:** How do we solve these equations?

Write as a linear system

$$\mathbf{Au} = \mathbf{b}$$

where A has the same sparsity structure as from RD diffusion. Also, its direct solution is expensive, even when accounting for sparsity structure.

- Can't us ADI, since (apparently) we can't factor operator into (1-m)(1+m)
- A: Use method of relaxation



1. plus shape: Instanteously satisfy equation for centerpoint
2. 2nd one: Move right and satisfy the equation for the next point. Will unsatisfy the previous point in the process!
3. large rectangle: Keep going with kick, we get a good solution despite the unsatisfactions (Gauss used this)

In his words

- Visit each unknown in grid in some order, (row by row, column by column).
  Fix/Freeze other unknowns, adjust our unknown so that its equation is 'instantaneously' satisfied.
- One complete pass through grid = a relaxation sweep

Under certain conditions, this solution will converge.
2 Main ways to update

1. Jacobian - use unknowns from previous iteration to satisfy local equations.
2. Gauss-Seidel - use unknowns most recently computed iteration

We will go over Gauss-Seidel.

- Faster
- better storage efficiency
- basis for successive over relaxation

Will also go over way that generalizes immediately to nonlinear equations
Let:

$$u_{i,j}^{(n)} = \text{discrete solution approx to } u_{i,j} \text{at nth iteration (sweep)}$$

here we assume i sweeps most rapidly

```
for j= 1:n
    for i = 1:n % not those exact indices since only sweeping interior points
```

Residual

$$r_{i,j}^{(n)} = h^{-2}(u_{i-1,j}^{(n+1)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n+1)} + u_{i,j+1}^{(n)} - 4u_{i,j}^{(n)}) - f_{i,j}$$

Updates i (right) direction first and then j (up) direction second. This explains the different n-indices in the residual equation.

November 17th 11am

Model Problem (ignoring B.C.'s)

$$u_{xx} + u_{yy} = f(x,y) \qquad u = u(x,y)$$

Residual for GS relaxation sweep

$$r_{i,j}^{(}n) = h^{-2}(u_{i-1,j}^{(n+1)} + u_{i+1,j}^{(n)} + u_{i,j-1}^{(n+1)} + u_{i,j+1}^{(n)} - 4u_{1,j}^{(n)}) - f_{i,j}$$

- Write relaxation in terms of update $\delta u_{i,j}^{(n)}$ and in a way that immediately generatlizes to nonlinear case.

$$u_{i,j}^{(n)} \rightarrow u_{i,j}^{(n+1)} = u_{i,j}^{(n)} - r u_{i,j}^{(n)}$$

- First define

$$F_{i,j}^{h} = \frac{u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1} - 4u_{i,j}}{h^2} - f_{i,j}$$

- Newton's Method

$$u_{i,j}^{(n+1)} = u_{i,j}^{(n)} - r_{i,j}^{(n)} \left[\left. \frac{\partial F_{i,j}^{h}}{\partial u_{i,j}} \right|_{u_{i,j}=u_{i,j}^{(n)}} \right]^{-1}$$

$$u_{i,j}^{(n)} - \frac{r_{i,j}^{(n)}}{-4h^{-2}}$$

$$= u_{i,j}^{(n)} + \frac{1}{4} h^2 r_{i,j}^{(n)}$$

- Equation above is equivalent to solving other equation directly for $u_{i,j}$ (verify)

11.6 Convergence of Relaxation Methods

- Collect all unknowns $u_{i,j}$ into single vector $\mathbf{u}$ of length $\approx N = n^2$ where n is the number of grid points per grid edge
- Write FD equations in matrix form

$$\mathbf{A}\mathbf{u} = \mathbf{b}$$

- Matrix $\mathbf{A}$ is diagonally dominant if

$$|a_{i,j}| \geq \sum_{j=1, j=i}^{N} |a_{i,j}|, \qquad i = 1, 2, \ldots, N$$

- Rough rule of thumb: GS converges if $\mathbf{A}$ is diagonally dominant (is for current case)
- How do we monitor convergence?
- Two Natural Approaches
  1. Compute Residual Norm: $||mathbf{r}^{(n)}||$
  2. Compute Solution update norm: $||mathbf{u}^{(n+1)} - mathbf{u}^{(n)}||$
- Check so called 'running residual' $\Rightarrow$ residuals that appear as we sweep through the mesh

- Consider iterative procedure:

$$\mathbf{u}^{(0)} \to \mathbf{u}^{(1)} \to \ldots \to \mathbf{u}^{(n)} \to \mathbf{u}^{(n+1)} \to \ldots \to \mathbf{0}$$

- Errors

$$\mathbf{e}^{(0)} \to \mathbf{e}^{(1)} \to \ldots \to \mathbf{e}^{(n)} \to \mathbf{e}^{(n+1)} \to \ldots \to \mathbf{0}$$

- For linear relaxation, view transformation or error $\mathbf{e}^{(n)}$ in terms of matrix $\mathbf{G}$ called the error amplification matrix

$$\mathbf{e}^{(n+1)} = \mathbf{G}\mathbf{e}^{(n)} = \mathbf{G}^2\mathbf{e}^{(n-1)} = \ldots \mathbf{G}^n\mathbf{e}^{(0)}$$

- Asymptotically, convergence is determined by spectral radius $\rho(\mathbf{G})$ of $\mathbf{G}$ where

$$\rho(\mathbf{G}) = \text{MAX}|\lambda_i(\mathbf{G})|$$

  where λ are the eigenvalues of $\mathbf{G}$

- Asymptotic convergence rate R

$$R \equiv \log_{10}(\rho^{-1})$$

- Interpolation: $1/R$ = the number of sweeps needed asymptotically to decrease $||\mathbf{e}^{(n)}$ by order of magnitude

- For Gauss-Siedel on model Probelm

$$\rho(\mathbf{G}_{\text{GS}} = 1 - O(h^2)$$

$$\Rightarrow 1/R = O(n^2) \Rightarrow \text{\# of sweeps for } ||\mathbf{e}^{(n)} \text{ to decrease by 10}$$

  $\Rightarrow$ total cost $\approx O(N^2) = O(n^4)$

- Verify slow convergence; G.S. not useful in practice

11.7 Test Solution

- Strategy: specify $u(x, y)$ which satisfies boundary conditions; then compute corresponding RHS function $f(x, y)$

$$u(x, y) = \sin(\omega_x x)\sin(\omega_y y)$$

  for ω_x and ω_y that are integer multiples of π

- Then

$$f(x, y) = -(\omega_x^2 + \omega_y^2)u(x, y)$$

- Supply f(x,y) to solution algorithm, should get $u(x, y)$ back (up to truncation error)

11.8 Successive Over Relaxation (SOR)

- Historically, researchers found that GS could be sped up by systematically 'overcorrecting' updated solutions valves

  $\Rightarrow$ SOR

- For model Probelm we have

$$u_{i,j}^{(n+1)} = \omega \hat{u}_{i,j}^{(n+1)} + (1-\omega) u_{i,j}^{(n)}$$

  where u hat is a solution from regular G.S.

$$1 < \omega < 2$$

  for stability

- Under ideal conditions, SOR reduces number of sweeps to $O(n)$. Total cost $O(n^3) = O(N^{3/2}$
- When ρ_{GS} can be computed exactly, can compute ω_{OPT} for model problem

$$\omega_{OPT} = \frac{2}{1 + \sin(\pi h)}$$

12 Stochastic (random) methods

12.1 Overview

Motivation

- Many problems in physics have a non-deterministic or random or stochastic character
- Canonical example: Brownian motion; random walk - dynamics of particle subject to some random force
- Many other examples: includes almost all statistical mechanical systems
- Simulation is often impeded by the need to compute (accurate) expectation values of physical quantities
- Consider random walk

$$r^2(t^n) \rightarrow \langle r(t^n) \rangle$$

  where $\langle \ldots \rangle$ is average over **many** trials
- sometimes qualitative info can be extracted from small number of simulations

Coverage

- Uniform random number generators (RNGs)
- Non-uniform random number generators
- Applications
  - Diffusion limited Aggregation
  - Ising Model

12.2 Pseudo-Random Number Generation (uniform)

- Almost all RNG in computer languages are algorithmic (deterministic); hence terminology 'pseudo' RNG (PRNG)
- A PRNG can be characterized by a probability distribution function (PDF), p(x) such that

$$p(x)dx \Rightarrow \text{probability that randomly generated number lies in internal x to x+dx}$$

- p(x) must be normalized on domain of interest $[x_{min}, x_{max}]$

$$\int_{x_{min}}^{x_{max}} p(x)dx = 1$$

- Caveat: floating point world is not the continuum, but we are going to ignore that point

- p(x) for uniform PRNG

- Many PRNG algorithms have been developed
- MATLAB provides 9; see held for rng
- One common choice, **Linear Congruential Generator**
- Generates sequences of integers whose bit pattern can be interpreted as reals in the range [0,1]

$$I_{n+1} = (aI_n + c) \text{mod} m$$

  where a and c are other integers and m is an integer called modulus

- Good choices for a,c will give maximal period m

Seeding a PRNG

- MATLAB by design, will return same random number at start-up
- Change this behaviour by seeding the RNG. Forces the RNG to start somewhere else in it's sequence
- MATLAB

```
rng(seed)

% where seed is a non-negative integer
% same sequence always generated for same seed
```

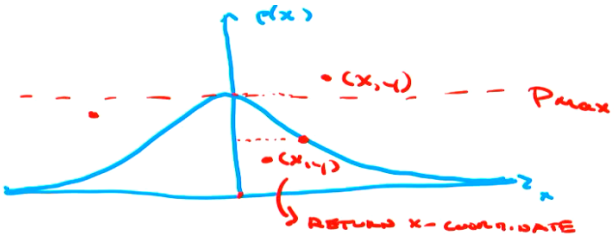
12.3 Pseudo-Random Number Generation (non-uniform)

- Consider some variable (deviate) which is randomly but not necessarily uniformly distributed in some interval $x_{\text{min}} \leq x \leq x_{\text{max}}$

  Examlpe: Gaussian Distibution $(-\infty < x < \infty)$

$$\text{PDF: } p(x) = \frac{1}{\sqrt{\pi\sigma}} e^{-x^2/\sigma^2}$$

- Many other examples: poisson, Maxwellian, ...
- Given $p(x)$, x_{min}, x_{max} and a [0,...1] uniform PRNG such as MATLAB's rand, how can we generate deviates distributed according to $p(x)$?



- Generate random points (x,y) [both coordinates randomly generated] in the region

$$x_{\min} \leq x \leq x_{\max}$$

$$y \leq p_{\max}$$

$$p_{\max} = max(p(x))$$

$\Rightarrow$ Rejects points that lie above $p(x)$, accepts points below p(x), returning x as the deviate

- Pseudo code for non-uniform PRNG

```
accept = false
until accept do
    x = random(xmin, xmax)
    y = random(0,ymax)
    if y<p(x) then rand = x
    accept = true
    end if
end do
```

Testing Implementation

- Generate large number, n, of deviates on [xmin, xmax]
- Compute approximation of PDF p(x) by "binning" deviates; i.e. count number of deviates in each interval/bin

$$x_i \text{ to } x_i + \Delta x$$

$$x_1 = x_{\min}$$

$$x_N = x_{\max}$$

$$\Delta x = \frac{x_{\max} - x_{\min}}{N_1}$$

$$N - 1 = \text{ \# of bins}$$

- Normalization (verify) let c_i be the count for the i-th bin, then

$$\bar{c}_i = \frac{c_i}{n\Delta x}$$

And

$$lim_{n\to\infty, N\to\infty}\bar{c}_i = p(x_i)$$

12.4 Diffusion Limited Aggregation (DLA)

Model for growth of clusters $\Rightarrow$ collections of identical particles
Examples:

1. Snow Flakes
2. Soot particles
3. Frost (Hoar Frost)
4. Lightning; sparks
5. Urban growth

- These clusters (aggregates) are characterized by dendritic (branching) structure
- Typically have non-integer dimensionality (fractal)

$$m(r) = \text{mass contained within radius r}$$

$$m(r) = r^d \rightarrow \text{fractal dimension}$$

$$\text{Typically} \quad 1 < d < 2$$

DLA Algorithm

- Use 2-D uniform lattice in (x,y)
  Particles: unit mass; two types
  - Fixed/ Frozen: structure at some specific lattice site
  - Mobile: free to random walk on lattice

  Discrete time steps t^n
  Simplest Implementation
  - One free particle at any time
  - Particle random walks until becomes adjacent to a lattice site with fixed particle
  - Free particle gets frozen at current position
  - New random walker is launched
  - Cluster: all fixed particles

Growth is **SLOW**

$$D_{\text{RMS}} = \langle D^2 \rangle^{\frac{1}{2}} \tilde{n}^{\frac{1}{2}}$$

where n is the number of steps

Extension: central bias

$$0 \leq b \leq 1$$

b : Probability that walker takes extra step directly toward the center

- Using the bias we can speed up the computation but it makes the fractal more clumpy and concentrated at the center.

missed friday class
November 27th 2023 11:06am (caught up)
Recall: Monte Carlo Method (ISING system)

```
if E_flip <= 0
    flip spin
else
    generate r = random(0,1)
    if re = exp(-E-flip((kBT)))
        flip spin
    else
        leave spin as is
    end
end
```

$$\Rightarrow \frac{P_1}{P_2} = \exp[-(E_1 - E_2)/(k_B T)]$$

$\Rightarrow$ what is needed for thermal equilibrium?

12.5 Physical Behaviour of Model

- System characterized by **competition**
  - Ferromagnetic interaction tends to **order** system (spin alignment favoured as $T \to 0$)
  - Temperature tends to disorder system (spin completely randomized at high T)
- At some critical temperature (T_c) and in the thermodynamic limit ($N \to \infty$) there will be a phase transition

Physical Quantities to measure (expectation values)

- When simulating, need to give system time to equilibrate; then run for long enough to generate "good statistics"
- Make passes through lattice; each pass generates new microstate α; compute expectation values by averaging over these states

- Typically normalize quantities by dividing by number of spins
  - Average energy $\langle E \rangle$
  - Avergae magnetization $\langle M \rangle$

$$E = -J \sum_{\langle ij \rangle} s_i s_j (-\mu H \sum_i s_i$$

- At T=0 (Take J=1); what is $\langle E \rangle$?

$$\langle E \rangle = -2J = -2;$$

  each spin contributes -4J; Divide by 2 to account for double-counting

- As $T \to \infty$, what is $\langle E \rangle$?

$$\langle E \rangle \to 0;$$

  But have to go to quite high T to see this since correlations among spins persist for even high T

Derived Quantities

- Consider Variance of Energy

$$(\Delta E)^2 \equiv \langle E^2 \rangle - \langle E \rangle^2$$

- Fluctuation-Dissipation theorem (connects fluctuations to response of system to external perturbation). Relates to specific heat.

$$C = \frac{(\Delta E)^2}{k_B T^2}$$

  Will work in units where $k_B = 1$

- Similarly, variance in M is related to susceptibility $\chi = \frac{dM}{dH}$ ($H \equiv$ Magnetic field)

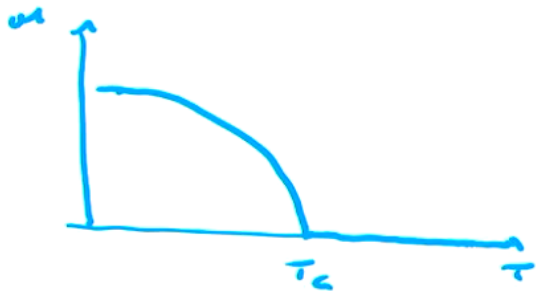
$$\chi = \frac{(\Delta M)^2}{k_B T}$$

  Behaviour near $T = T_c$

- Scaling of Magnetization

$$M \sim (T - T_c)^\beta$$

  $\beta \to$ universal exponent



- exact solution of 2D ISING model (onsager 1944)

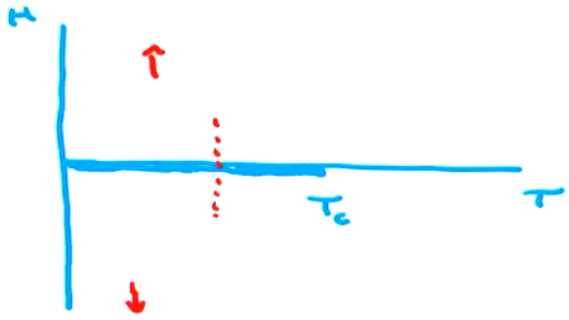
$$T_c = \frac{2}{ln(1+\sqrt{2}} = 2.2692\ldots$$

$$\beta = \frac{1}{8}$$

- C and χ are divergent at $T = T_c$
- Phasee transition for $M = 0$ is called second order or continuous since the order parameter $\langle M \rangle$ does not have a jump at $T = T_c$

12.6 Behaviour of Model for $H \neq 0$

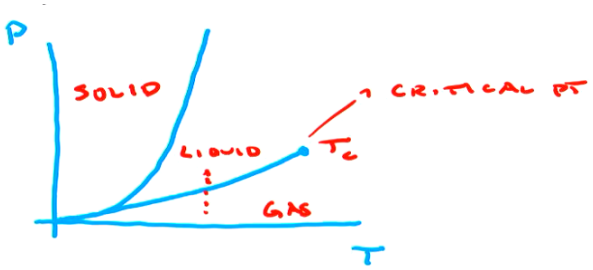
- Now have 2 extensive parameters: T, H
- Phase Diagram
  - For low T, $T < T_c$, per spin magnetization is < 0 on one side and > 0 on other



- Thus, have a discontinuity in $\langle M \rangle$ as we tune across H=0

- $\Rightarrow$ first order phase transition
- Line in T-H plane
- Above T_c there is no spontaneous magnetization so no possibility of discontinuity in $\langle M \rangle$ as H passes through 0
- Phase transition line terminates at critical point
  $\Rightarrow$ general feature of first-order phase transition

12.7 Phase Transitions



- Similar situation: Gas-Liquid System: First order over a range of pressures, with a discontinuity in density
- As P increase, size of discontinuity decreases, vanishes at critical point
- Key difference between second order, first order phase transitions
  - Second order: fluctuations become large near transition.
  - First order: No growth in fluctuation near transition; happens suddenly
- $\langle M \rangle$ vs H for ISING system can exhibit hysteresis, especially at low T

13 Fourier Transform

(following Numerical Methods, Press et al.)

13.1 Quick Review

- Transform pairs: $h(t) \Longleftrightarrow H(f)$

$$h(at) \Longleftrightarrow \frac{a}{|a|} H(\frac{f}{|H|}$$

- Convolution

$$g * h \equiv \int_{-\infty}^{\infty} g(\tau) h(t-\tau) d\tau$$

$$g * h \Longleftrightarrow G(f)H(f) \quad \text{Convolution Theorem}$$

- Total Power

$$\int_{-\infty}^{\infty} |h(t)|^2 dt = \int_{-\infty}^{\infty} |H(f)|^2 df$$

  Parseval's Theorem

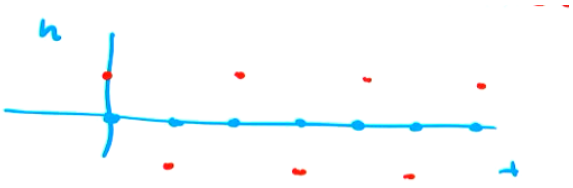
13.2 FT of Discretely Sampled Data

13.2.1 Sampling Theorem and Aliasing

- Assume $h(t)$ sampled uniformly in time

$$h_n = h(\Delta n), \quad n = \ldots, -1, 0, 1, \ldots$$

- Sampling Rate: $\frac{1}{\Delta}$
- Nyquist Critical Frequency: $f_c = \frac{1}{2\Delta}$



- Sampling Theorem: If continuous function h(t), samples at interval Δ, its bandwidth limited to frequencies smaller than f_c ($H(f) = 0$ for all $|f| \geq f_c$) then function is completely determined by its samples h_n

$$h(t) = \Delta \sum_{n=-\infty}^{+\infty} h_n \frac{\sin(2\pi f_c(t - n\Delta)}{\pi(t - n\Delta)}$$

- What happens if function is not bandwidth limited?
- All power spectral density outside nyquist range $-f_c < f < f_c$ is moved into that range $\Rightarrow$ **Alisasing**.
- Two waves $\exp(2\pi f, t)$ and $\exp(2\pi f_c t)$ yield same samples if and only if f, and f_c by multiple of $\frac{1}{\Delta} \rightarrow$ width in frequency of range $(-f_c, f_c)$

13.2.2 Discrete Fourier Transform

- Now what to estimate Fourier transform if a function from finite number of sample points.
- Suppose we have

$$h_k \equiv h(t_k), \qquad t_k = k\Delta, \quad k = 0, 1, \ldots, N-1$$

  Where *Delta* is the spacing and N is the number of samples.
- Will assume that N is even
- Seek estimates at discrete values

$$f_n = \frac{n}{N\Delta}, \qquad n = \frac{-N}{2}, \ldots, \frac{N}{2}$$

- Extreme values $\rightarrow$ lower and upper limits of the Nyquist critical frequency range
- Estimates for $n = -N/2$, $N/2$ are equal $\Rightarrow$ there are a total of N independent H_n

$$H(f) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f t} dt$$

- Approximate using discrete sum

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} \approx \sum_{k=0}^{N-1} h_k e^{2\pi i f_n t_n} \Delta \tag{13.1}$$

$$= \Delta \sum_{k=0}^{N-1} h_k e^{2\pi i k_n / N} \tag{13.2}$$

- Defines discrete Fourier Transform

$$H_n = \sum_{k=0}^{N-1} h_k e^{2\pi i k_n / N} \tag{13.3}$$

- Have

$$H(f_n) \approx \Delta H_n$$

- n value from -N/2 to N/2
- equation above is periodic in n with period N

$$\Rightarrow H_{-n} = H_{N-n}, \qquad n = 1, 2, \ldots$$

- Useful convention, let n take on values 0,1,2,..., N-1
  - Zero frequency: n=0
  - Positive frequency: n=1,2,...,N/2-1
  - Negative frequency: n=N/2+1,N/2+2,...,N-1
    For Total of N samples
- Discrete inverse Fourier transform

$$h_k = \frac{1}{N} \sum_{n=0}^{N-1} H_n e^{-2\pi i k_n / N}$$

- Discrete version of Parseval's Theorem

$$\sum_{k=0}^{N-1} |h_n|^2 = \frac{1}{N} \sum_{n=0}^{N-1} |H_n|^2$$

13.2.3 Fast Fourier Transform

- How much computation does it take to evaluate equation (73) above?
- Define complex number

$$w = e^{2\pi i / N}$$

- (73) can be written as

$$H_n = \sum_{k=0}^{N-1} w^{nk} h_k$$

  (exponent gives: nk-th power of w)
- In vector-matrix form

$$\mathbf{H} = \mathbf{A}\mathbf{h} \tag{13.4}$$

$$A_{ij} = w^{ij}$$

- (74) take N^2 multiplications and somewhat fewer operations to compute all w^{ij}
- Can actually evaluate FT in $O(N \ln N)$ operations $\Rightarrow$ Fast Fourier Transform $\Rightarrow$ FFT
- Key observation (Danielson; Lankzos in 1942, but many others before, including Gauss)

$$F_k = F_k^{\mathrm{e}} + w^k F_k^{\mathrm{o}}$$

$$e \equiv \text{even}$$
$$o \equiv \text{odd}$$

  Can be applied recursively to F_k^e, F_k^o all the way to single samples if N is a power of 2.